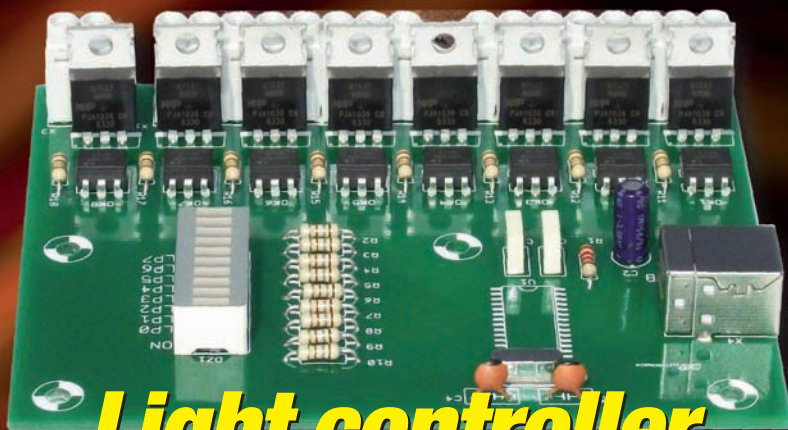
MICROCONTROLLORI Visione artificiale Display Dotmatrix
CONVERSIONE D-A PROGAMMAZIONE PIC PASCAL PWM
Controllo luci PIC18 USB Raspberry PI MIKROBASIC

fare elettronica

www.farelettronica.com
330 Dicembre Mensile Anno XXVIII € 6,00

Light controller USB

LE ALTERNATIVE ALLA MEMORIA FLASH

DEMOBOARD PER RASPBERRY PI

COMUNICAZIONE 1-WIRE CON PIC

DEBOUNCING IN MIKROPASCAL

REGOLATORE DI VELOCITÀ CON ATtiny24

COMUNICAZIONE MASTER-SLAVE CON PIC18

ISSN 1591-2272

330 *dicembre 2012*

Zoom in

Progettare & costruire

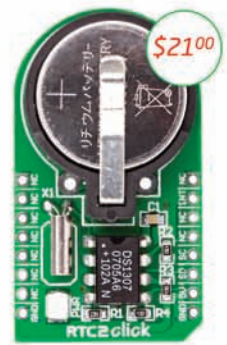
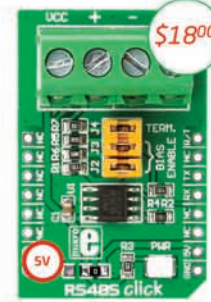
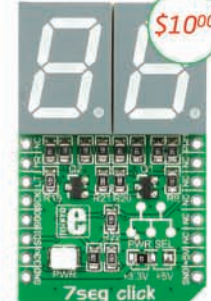
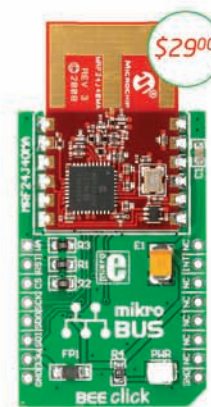
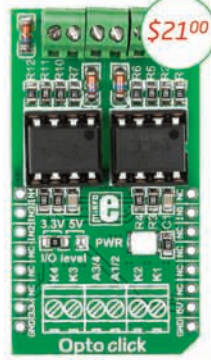
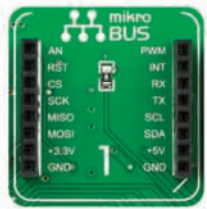
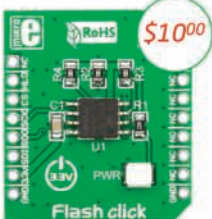
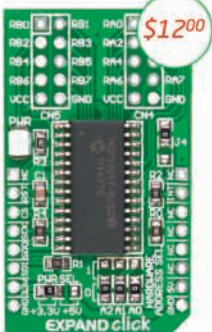
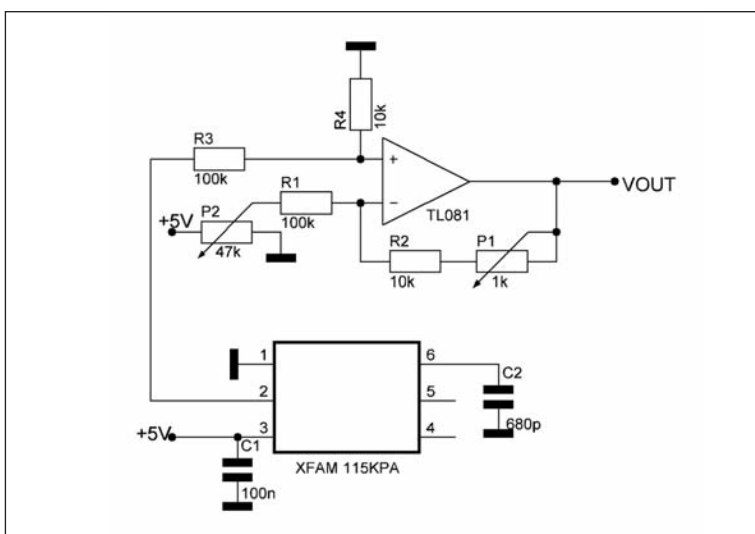
Rispondi e... **VINCI!** *pag.* **32**

DIVERTITI E METTI ALLA PROVA LE TUE CONOSCENZE CON **ELETTRO QUIZ** E VINCI OGNI MESE ESCLUSIVI PREMI!

di MAURIZIO DEL CORSO

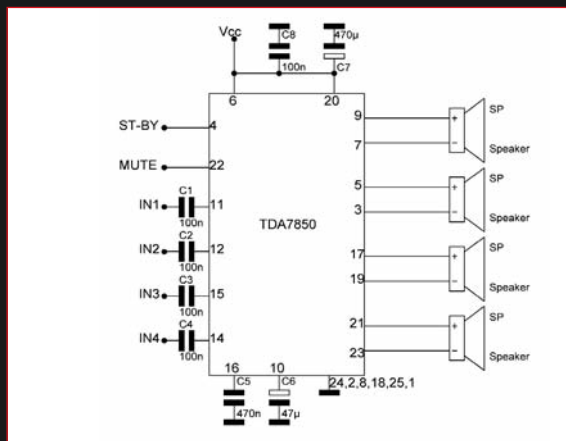
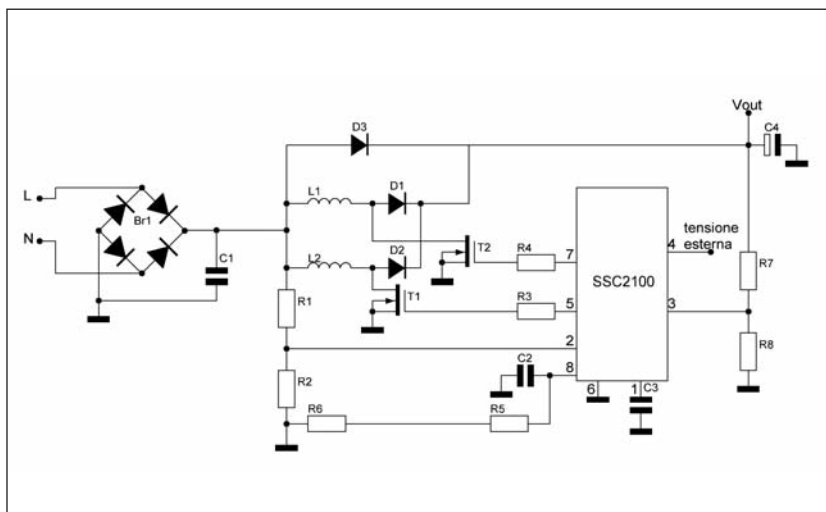
Toccare con mano

A gennaio incontriamoci a Modena Makers, nell'ambito di Expoelettronica, per toccare con mano e avere nozioni sulle stampanti 3D e i droni volanti.

Chi lavora in un certo campo specialistico, è facile che incorra nell'errore di credere che tutti siano sufficientemente eruditi da comprenderne i termini e le nozioni con estrema facilità. Per una rivista di elettronica (ma il concetto è estendibile in termini generali) il presupposto di partenza deve essere il completo opposto: elargire i contenuti in modo che siano fruibili da tutti, cercando il più possibile di non dare mai niente per scontato. Nei numeri passati, abbiamo affrontato il tema delle stampanti 3D, in questo numero parliamo di schede elettroniche in grado di far volare autonomamente degli aeromodelli… quanti non hanno mai visto dal vero una stampante 3D al lavoro? Quanti non hanno mai visto alzarsi in volo un quadricottero comandato da una scheda intelligente? Dunque per tutto coloro che vogliono toccare con mano, diamo appuntamento il 19 e 20 gennaio prossimi presso la fiera dell'elettronica di Modena dove mostreremo gli ultimi ritrovati nel campo delle stampanti 3D e dei droni volanti e dove potrete seguire le sessioni formative tenute dagli esperti del settore. Ovviamente è anche una buona occasione per conoscerci personalmente.
Vi aspetto!

SENSORI DI PRESSIONE ATMOSFERICA

La serie degli integrati XFAM-115KP è utilizzata per rilevare la pressione atmosferica. XFAM-115KP consente di ottenere in uscita una tensione lineare proporzionale alla pressione atmosferica, utilizzando il circuito di condizionamento in figura. In particolare fornisce una tensione da 0 V a 5 V con andamento lineare, in un range di pressione compreso tra 15 e 115 kPa.

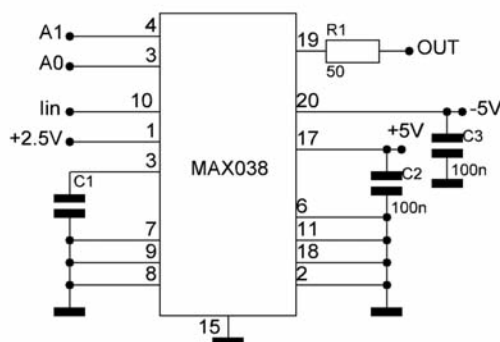
STABILIZZATORE DI TENSIONE

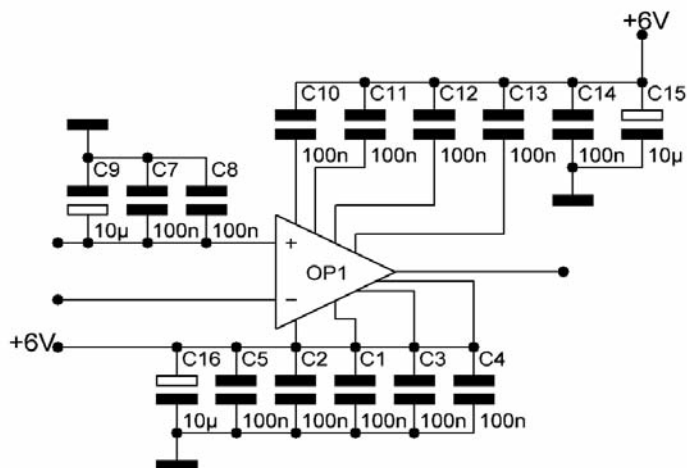
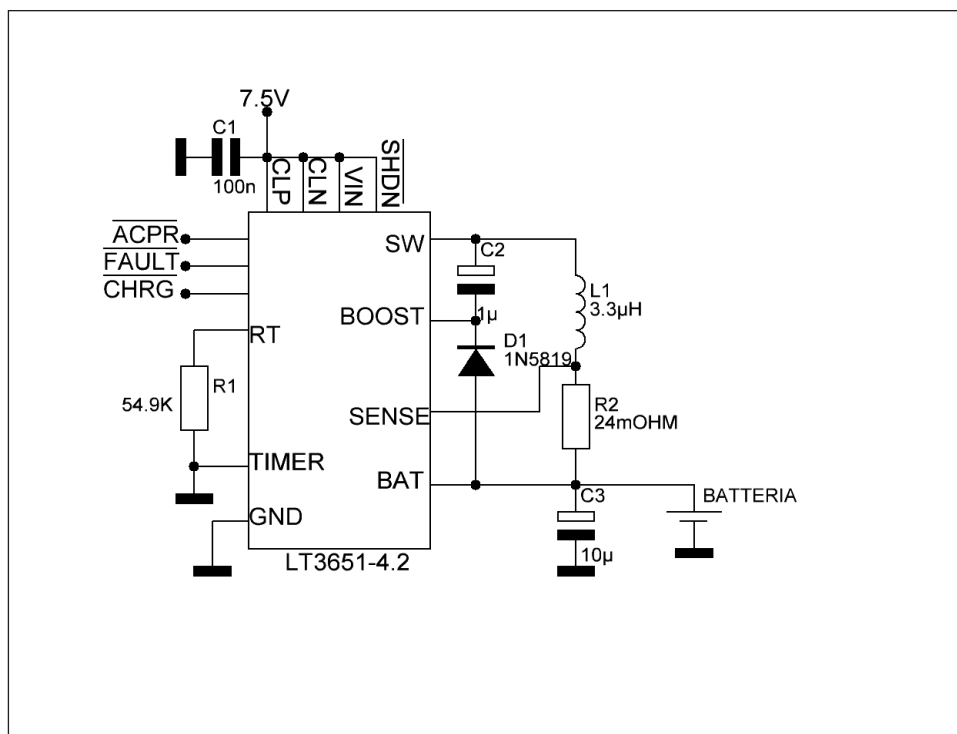
I dispositivi della serie SSC2100 sono chip per la stabilizzazione della tensione continua ottenuta dai 220 V. Vengono impiegati in elettrodomestici come televisori, computer ecc. Il loro vantaggio è quello di avere un basso costo e alte prestazioni per quanto riguarda il Ripple della tensione di uscita, il basso rumore e la dissipazione di corrente. Viene utilizzato per il ridotto impiego di componenti esterni.

Amplificatore in classe AB

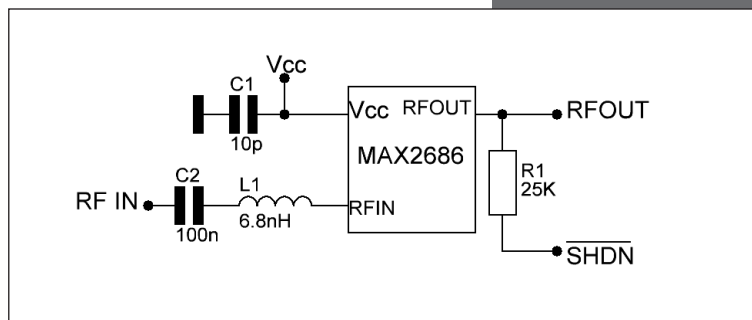
Il TDA7850 è un amplificatore di classe AB che integra al suo interno dei MOSFET. L'integrato si presenta in un package del tipo Flexiwatt 25, pacchetto progettato per autoradio di alta potenza. L'utilizzo di MOSFET complementari garantisce un minor spreco di corrente. Può essere alimentato con un range di tensione da 10 V a 28 V. Alimentato a 14 V fornisce in uscita una potenza di 60 W.

Generatore di segnali AF

Il MAX038 è un generatore in grado di riprodurre diversi segnali in alta frequenza, ad esempio forme d'onda a dente di sega, sinusoidale, quadra. Il valore della frequenza può essere variato da 0,1 Hz a 20 MHz, la tensione di alimentazione è di 2,5 V.

Le forme d'onda sono selezionabili tramite un dip switch a due bit. L'uscita a bassa impedenza può pilotare fino a ± 20 mA. L'oscillatore interno riesce a mantenere un duty cycle stabile dello 0,5.

Ampli 2 W a 4 stadi

Il HMC906 è un quattro stadi di GaAs MMIC pHEMT 2 Watt Power Amplifier che opera tra 27,3 e 33,5 GHz. Il HMC906 fornisce 23 dB di guadagno e 34 dBm di potenza di uscita saturi e il 22% PAE da un alimentatore 6 V. La RF I/O è DC bloccata e abbinata a 50 Ohm per la facilità di integrazione in Multi-Chip-Module (MCM).

AMPÈRE IN USCITA

LT3651 è un regolatore che accetta in ingresso tensioni da 4 V a 32 V, offrendo 4 A in uscita. È stato realizzato per essere utilizzato nei caricabatterie al litio. Un ciclo di ricarica inizia con l'inserimento della batteria o quando la tensione della batteria scende sotto il 2,5% del valore nominale. Il timer è impostato a tre ore ed è programmabile dall'utente. Una volta che la ricarica è terminata, la corrente dell' LT3651 scende a 85 µA in modalità standby.

AMPLI A BASSO RUMORE

Il MAX2686/MAX2688 è un amplificatore a basso rumore (LNA) per applicazioni GPS L1, Galileo, GLONASS. È progettato per raggiungere un elevato guadagno e ultra-low-noise. Il MAX2686 fornisce un guadagno di 19 dB. I dispositivi operano da un 1,6 V a 3,3 V e sono disponibili in un package di dimensioni molto ridotte (0,86 x 0,86 x 0,65 mm).

UEP 50: Programmatore Universale di Grifo

L'UEP-50 è un programmatore universale che gestisce tutti i tipi di dispositivi programmabili. La caratteristica di avere tutti i piedini pilotati separatamente da una opportuna logica consente l'impostazione di: livelli logici, pull up o pull down, clock, massa, una tensione di alimentazione e due tensioni di programmazione oltre a, naturalmente, consentire la lettura indipendente dello stato di ognuno dei 48 piedini. Questo avanzato progetto fornisce la possibilità di programmare quasi tutti i dispositivi in un contenitore DIL (doppia fila) fino a 48 piedini, senza adattatori e senza moduli speciali aggiuntivi. Il supporto degli attuali dispositivi programmabili, e dei futuri che usciranno, fornisce agli utilizzatori la libertà di scegliere sempre il miglior dispositivo per ogni progetto realizzato o da realizzare.

L'UEP-50 è distribuito da Grifo - Via dell'Artigiano 8/6 - 40016 San Giorgio di Piano (BO).

CODICE MIP 2830725

IGBT Toshiba da 600 V di sesta generazione

Toshiba Electronics Europe ha annunciato gli IGBT di sesta generazione in grado di offrire un migliore compromesso tra perdite di commutazione e perdite di conduzione con una maggiore efficienza e migliori prestazioni. La nuova tecnologia di sesta generazione è alla base di una nuova famiglia di dispositivi compatti da 600 V adatti per una varietà di sistemi a commutazione "brusca" (hard switching), tra cui azionamenti, inverter fotovoltaici e gruppi di continuità (UPS, Uninterruptible Power Supplies).

Gli IGBT Toshiba di sesta generazione combinano una progettazione più fine con un processo di lavorazione dei wafer di tipo "punch-through" più sottile rispetto alla precedente generazione e presentano uno sviluppo verticale altamente ottimizzato.

I nuovi prodotti di sesta generazione sono caratterizzati da correnti nominali di 15 A (GT15J341), 20 A (GT20J341), 30 A (GT30J341) e 50 A (GT50J342). Ciascun componente comprende un IGBT e un diodo a recupero inverso rapido tra emettitore e collettore, inseriti entrambi in un unico contenitore compatto.

Tutti i dispositivi presentano una VCE(sat) tipica di 1,5 V alla corrente nominale.

CODICE MIP 2830963

Mouser offre gli MCU a segnale misto STM32 F3 di STMicroelectronics

Mouser Electronics, Inc. ha annunciato la disponibilità a magazzino degli MCU serie STM32 F3 di STMicroelectronics che combinano il core ARM Cortex-M4 a 32 bit con un DSP, istruzioni FPU e periferiche analogiche avanzate.

La serie STM32 F3 rappresenta un'innovazione del DSC che combina il core Cortex-M4 con 5 veloci MPSP a 12 bit, precisi ADC sigma-delta a 16 bit, amplificatori di guadagno programmabili (4 guadagni, precisione dell'1%), veloci comparatori da 50 ns e versatili unità di controllo tempo a 144 MHz, per un'integrazione ottimale. I dispositivi sono ottimizzati per la manipolazione e l'elaborazione efficienti dei segnali misti nei circuiti quali controller per i motori trifase, biometria e uscite di sensori industriali o filtri audio, consentendo ai progettisti di gestire le applicazioni di controllo a segnale misto.

Gli MCU serie STM32 F3 a segnale misto sono compatibili a livello di pin con la serie STM32 F1 e la nuova serie STM32 F3 arricchisce il portafoglio Cortex-M4 STM32 per offrire sia costi entry level con la serie F3, sia prestazioni massime con la serie F4. La serie STM F3 include dispositivi con 64-256 Kbyte di memoria Flash on chip, fino a 48 Kbyte di memoria SRAM.

CODICE MIP 2831075

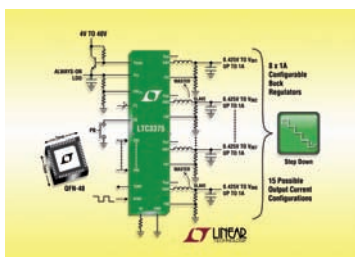
LabVIEW 2012 dà un impulso al successo e migliora la scalabilità dei sistemi di misura e controllo

National Instruments ha presentato NI LabVIEW 2012, l'ultima versione dell'unico software di progettazione di sistemi per tecnici e ingegneri. Gli utenti ricevono una nuova serie di funzioni pronte all'uso per una vasta gamma di applicazioni LabVIEW e l'accesso a opzioni formative, che consentono di migliorare la qualità dei propri sistemi. Queste nuove caratteristiche testimoniano il continuo impegno di National Instruments nel fornire una piattaforma che dia un impulso al successo in qualsiasi sistema di controllo e misura; e assicuri agli utenti la capacità di innovare in tutta sicurezza. Funzioni di LabVIEW 2012:

- template e progetti d'esempio;
- corsi online di autoapprendimento;
- stabilità ottimizzata;
- nuovi strumenti per analisi ad alte prestazioni ed elaborazione immagini avanzata;
- produttività ottimizzata grazie alla comunitià degli utilizzatori;
- app mobili per la visualizzazione e il controllo su iPad.

CODICE MIP 2830782

Regolatori DC/DC buck a otto canali da 1 A, configurabili per sistemi multi-rail

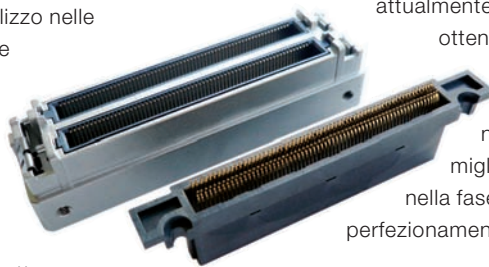
Linear Technology Corporation presenta l'LTC3375, una soluzione di power management generica altamente integrata per i sistemi che richiedono più rail di alimentazione a bassa tensione. Il dispositivo è dotato di 8 canali da 1 A indipendenti, con interfaccia di controllo I2C, sequencing flessibile e monitoraggio dei guasti, il tutto in un package QFN compatto. L'LTC3375 contiene otto regolatori sincroni step-down ad alta efficienza con compensazione interna, oltre a un controller LDO ad alta tensione always-ON da 25 mA. Ciascun regolatore buck possiede un'alimentazione indipendente da 2,25 V a 5,5 V e un range di tensioni di uscita compreso tra 0,425 V e VIN. Il controllo ON/OFF/RESET a pulsante, il reset all'accensione e il watchdog timer forniscono il sequencing di accensione flessibile e affidabile e il monitoraggio del sistema. L'LTC3375 è provvisto di un oscillatore programmabile e sincronizzabile da 1 MHz a 3M Hz con una frequenza di commutazione predefinita di 2 MHz. La corrente di riposo è di appena 11 µA con tutti i dispositivi DC/DC disattivati, il che assicura l'ottimizzazione della durata della batteria. Il dispositivo è ideale per una vasta gamma di applicazioni multicanale, inclusi i sistemi industriali, automotive e di comunicazione.

CODICE MIP 2831169

RS Components migliora il sito Internet

RS Components ha apportato numerosi e significativi miglioramenti al suo sito Internet per rendere più semplice, chiaro ed efficiente il processo di acquisto. I cambiamenti sono il risultato di quattro mesi di ricerche approfondite tra i propri clienti e consentono agli utenti di risparmiare tempo, fatica e denaro. È stata migliorata sia la gestione del profilo sia la consultazione dello storico ordini, permettendo ai clienti di consultare fino a 13 mesi di cronologia degli ordini effettuati attraverso qualsiasi canale (online, telefono, fax o punto vendita). Gli utenti potranno, inoltre, rintracciare la consegna del proprio ordine partendo dal codice prodotto, ripetere ordini direttamente dallo storico, inoltrare l'ordine a un collega e richiedere una copia delle fatture in formato elettronico. L'intero processo di acquisto è stato trasformato grazie a un nuovo layout, capace di fornire al cliente una più chiara e completa interfaccia, e all'introduzione di un migliore supporto attraverso FAQ e opzioni di contatto. All'interno del carrello i prezzi e le opzioni di consegna sono più chiaramente indicati e il costo totale dell'ordine è visibile durante tutto il processo di acquisto. Altra novità è la speciale funzione guest checkout che semplifica il processo di acquisto per i nuovi clienti.

CODICE MIP 2831157

Da ITT Cannon nuovo connettore Quad Lock

ITT Cannon ha presentato un nuovo connettore Quad Lock (QLC) con forza di inserzione zero (ZIF) per l'utilizzo nelle apparecchiature medicali di diagnostica e imaging. Il dispositivo costituisce una novità per l'industria, avendo ottenuto una riduzione dimensionale del pacchetto dell'interconnessione, mentre soddisfa le esigenze di salde prestazioni e ne incrementa le funzionalità, come l'accoppiamento mediante fissaggio a scatto. L'avere ridotto la spaziatura dalla linea DL standard a 0,8 mm ha permesso agli ingegneri di ITT ICS di progettare il sistema di connettore portatile ZIF QLC con una configurazione completa di 260 pin e di ridurre le dimensioni del pacchetto: è del 65% più piccolo dei prodotti DL esistenti e dei prodotti QLC concorrenti attualmente presenti sul mercato. Ciò permette ai clienti di ottenere una drastica riduzione delle dimensioni complessive delle apparecchiature diagnostiche. L'alto numero di 260 posizioni di pin offre numerose posizioni di canale e conferisce miglioramenti alla risoluzione dell'immagine possibile nella fase di disegno, che alla fine risulterà nel perfezionamento delle funzionalità diagnostiche.

CODICE MIP 2830937

Digi-Key Corporation e Fremont Micro Devices USA Inc. stipulano un accordo di distribuzione globale

Digi-Key Corporation e Fremont Micro Devices USA Inc., fornitore e leader a livello internazionale nel settore delle EEPROM seriali e dei prodotti a circuiti integrati per la gestione efficiente della potenza, hanno siglato un accordo di distribuzione globale. Le attività di FMD sono imperniate attorno allo sviluppo delle tre aree della memoria non volatile utilizzate nel settore delle memorie a bassa densità ed embedded. UltraEE, la tecnologia EEPROM brevettata da FMD, consente di realizzare le memorie a bassa densità con il miglior rapporto prezzo-prestazioni dell'intero settore. Negli ultimi otto anni, FMD ha acquisito una fetta sostanziosa del mercato delle EEPROM e si prevede che la crescita nei prossimi tre anni supererà il 40% su base annua. FMD offre memorie standard con densità da 2 a 1024 kB e memorie realizzate in base ai requisiti del cliente.
I dispositivi di gestione della potenza e le EEPROM di Fremont Micro Devices sono ora disponibili per l'acquisto sui siti web internazionali di Digi-Key.

CODICE MIP 2830753

Il principale salone mondiale alimenta la fiducia dell'industria elettronica

Con oltre 72.000 visitatori si è chiusa venerdì 16 novembre la venticinquesima edizione di Electronica, il salone internazionale di componenti, sistemi e applicazioni dell'elettronica. 2.669 espositori da 49 Paesi hanno mostrato, nell'arco di quattro giornate, il futuro dell'industria elettronica mediante una ricca offerta di soluzioni concrete. Al centro dell'attenzione soprattutto le tecnologie intelligenti ad alta efficienza energetica nelle aree di Energy Storage, LED e Smart Grid. A quest'ultimo tema è stata dedicata quest'anno anche la CEO Round Table, uno dei momenti salienti del programma collaterale.
Il settore guarda con moderato ottimismo al 2013, come ha dichiarato anche Christoph Stoppok, Direttore Generale delle associazioni di componentistica, sistemi elettronici e circuiti stampati all'interno della ZVEI (associazione nazionale tedesca dell'industria elettrotecnica ed elettronica): "Dopo una leggera contrazione, per il mercato mondiale dei componenti elettronici ci aspettiamo una crescita del quattro per cento circa, che porterà il giro d'affari a 480 miliardi di dollari. E le previsioni indicano un aumento del fatturato anche per l'industria dei semiconduttori." La prossima edizione di Electronica si svolgerà dall'11 al 14 novembre 2014 a Monaco di Baviera.

CODICE MIP 2831182

È arrivata la nuova EasyPIC Fusion 7!

Mikroelektronika, produttore di sistemi di sviluppo per le principali architetture a microcontrollore, schede di sviluppo, compilatori, schede accessorie, software e libri per microcontrollori e azienda, rappresentata in Italia da Inware srl, ha annunciato la nuova EasyPIC Fusion v7.

EasyPIC Fusion v7 è la nuova scheda di sviluppo di MikroElektronika che unisce il supporto per tre diverse architetture: dsPIC33, PIC24 e PIC32 in una singola scheda. Contiene molti moduli on-board, tra cui moduli multimediali, Ethernet, USB, CAN e altri. Inoltre, on-board il potente programmatore mikroProg e il debugger In-Circuit che supportano 65 microcontroller.

La scheda ha moduli per tutti i tipi di sviluppo. Lo schermo TFT con Touch Panel e il codec MP3 stereo consentono di sviluppare applicazioni multimediali. Sulla scheda ci sono due connettori USB e due connettori USB-UART. Il buzzer piezoelettrico vi aiuterà a testare il segnale del suono, ed è possibile utilizzare l'I2C EEPROM, la Serial Flash e lo slot per le schede microSD per memorizzare i dati. Sono presenti i socket per i sensori di temperatura DS1820 e LM35.

Inoltre sulla scheda sono presenti i socket MikroBUS: questo innovativo socket consente di utilizzare decine e decine di schede accessorie "Click" con estrema semplicità e senza alcuna regolazione hardware.

CODIE MIP 2831187

Maxim Integrated e RadioPulse per la realizzare soluzioni ZigBee ad alto grado di integrazione per smart grid

Maxim Integrated Products, Inc. ha annunciate la stipula di un accordo di collaborazione con RadioPulse finalizzato alla fornitura di funzioni di comunicazioni ZigBee per soluzioni smart grid. Insieme, le due società svilupperanno e commercializzeranno software per ZigBee, SoC (System on Chip) ad alto grado di integrazione e transceiver "smart", al fine di potenziare l'attuale offerta di soluzioni per smart grid di Maxim e aumentare le potenzialità di Zeus, il contatore "intelligente" (smart meter) integrato su SoC di recente introduzione. I primi frutti di questa collaborazione sono rappresentati da soluzioni per smart grid che si distinguono per il loro elevato grado di integrazione. La sinergia tra il software per comunicazioni wireless ZigBee di RadioPulse e le soluzioni "intelligenti" e sicure di Maxim basate su circuiti integrati permette ai clienti di ridurre sensibilmente tempi di sviluppo, ingombri sulla scheda e costi di realizzazione. I vantaggi di questa cooperazione sono immediatamente evidenti quando si prendono in considerazione le soluzioni attualmente disponibili, di tipo non integrato.

CODICE MIP 2831190

Power Integrations lancia la famiglia di IC con commutazione offline TinySwitch-4

Power Integrations ha presentato la famiglia TinySwitch-4, la nuova generazione della serie TinySwitch di circuiti integrati con switch offline leader di settore. Provvisti di protezione dalla corrente di sovraccarico con compensazione di linea, i dispositivi TinySwitch-4 riducono in modo significativo la potenza di sovraccarico massima che può essere fornita al carico sull'intero range di tensioni di ingresso AC. I dispositivi incorporano un MOSFET da 725 V che consente il facile derating fino all'80% per la tensione altamente riflessa e i progetti ad elevata tensione di bus. L'efficienza superiore al 70% e un carico al 5% consentono ai progettisti di soddisfare le rigorose specifiche europee per lo standby che dovrebbero entrare in vigore nel 2013, mentre il consumo di potenza senza carico inferiore a 30 mW a 230 VAC soddisfa i requisiti di risparmio energetico del mercato odierno.

CODICE MIP 2831194

GLI SWITCH BOOST PER APPLICAZIONI LED RETROILLUMINATE DI FAIRCHILD SEMICONDUCTOR

I progettisti di sistemi a LED ad alta potenza devono affrontare la sfida che consiste nel prevenire le perdite di potenza e le oscillazioni subarmoniche associate all'inefficienza dei convertitori boost. I controller boost single-channel FAN7340 e FAN73402 di Fairchild Semiconductor con MOSFET ad alta tensione e capacità di dimming integrata forniscono una risposta a questa sfida aiutando i progettisti a ottenere superiori livelli di efficienza, performance e affidabilità in sistemi d'illuminazione ad alta potenza quali pannelli LED retroilluminati per televisori e monitor 3D.
Questi switch boost retroilluminati utilizzano una topologia CMC con compensazione programmabile della pendenza (o coefficiente angolare) per prevenire le oscillazioni subarmoniche. Progettati con un controllo PWM analogico e di dimming interno, questi dispositivi superano i difetti caratteristici dei convertitori boost DCM fornendo una risposta di dimming PWM più rapida.
Entrambi i dispositivi forniscono funzioni di protezione integrate quali UVLO protezione LED aperto e protezione da sovratensioni OVP per contribuire a minimizzare il numero delle componenti necessarie. Questi dispositivi dispongono inoltre di una funzione per la prevenzione degli errori che genera un segnale FAULT con ritardo nel caso del verificarsi di condizioni anomale nella striscia LED.

CODICE MIP 2831192

Sei MAKER, SMART

Entra anche tu nel mondo dell'

€69.50

GENIUS MEMBERSHIP

- **11 nuovi numeri di Fare Elettronica sia in edizione cartacea che in PDF (in coda all'eventuale abbonamento esistente)**
- **11 nuovi numeri della rivista digitale firmware (in coda all'eventuale abbonamento esistente) tutto l'archivio delle riviste firmware pubblicate mensilmente a partire dal febbraio 2010**
- **accesso al CLUB della rivista Firmware con centinaia di articoli in PDF**
- **accesso al CLUB della rivista FareElettronica (centinaia di articoli in PDF)**
- **due ebook a scelta* (smartcard, CPLD, display, PIC, AmpOP, alimentatori, linguaggio C)**
- **due raccolte delle annate in PDF di Fare Elettronica (dal 2003 al 2011) e Firmware (dal 2006 al 2011) a scelta***

SCEGLI LA TUA MEMBERSHIP E FARAI PARTE ANCHE TU DEL CLUB PIÙ ESCLUSIVO DELL'ELETTRONICA!

WWW.IESHOP.IT/MEMBER

*la scelta viene effettuata al ricevimento della card

o GENIUS?

elettronica con Inware Edizioni!

MAKER MEMBERSHIP

Special Price €55.00

- 11 nuovi numeri di Fare Elettronica sia in edizione cartacea che in PDF (in coda all'eventuale abbonamento esistente)
- accesso al CLUB della rivista Fare Elettronica (centinaia di articoli in PDF)
- un ebook a scelta* (smartcard, CPLD, display, PIC,AmpOP, alimentatori, linguaggio C)
- una raccolta delle annate in PDF di Fare Elettronica a scelta* (dal 2003 al 2011)

€29.50

SMART MEMBERSHIP

- tutto l'archivio delle riviste firmware pubblicate mensilmente a partire dal febbraio 2010
- 11 nuovi numeri della rivista digitale firmware (in coda all'eventuale abbonamento esistente)
- accesso al club della rivista Firmware con centinaia di articoli in PDF
- un ebook a scelta* (smartcard, CPLD, display, PIC, AmpOP, alimentatori, linguaggio C)
- una raccolta della annate in PDF di Firmware (dal 2006 al 2011) a scelta*

CODICE MIP ???????

SI, VOGLIO RICEVERE SUBITO LA MIA MEMBERSHIP CARD

☐ MAKER MEMBERSHIP ☐ SMART MEMBERSHIP ☐ GENIUS MEMBERSHIP

METODI DI PAGAMENTO

☐ BONIFICO BANCARIO ☐ BOLLETTINO POSTALE ☐ ALLEGO ASSEGNO (Intestato a Inware Edizioni)

☐ MasterCard ☐ VISA ☐ postepay

PayPal fare il pagamento su paypal@inwaredizioni.it

TITOLARE..

NUMERO..

DATA DI SCADENZA........................ CODICE DI SICUREZZA..........

INVIARE IL MODULO D'ORDINE A: Inware Edizioni srl, via Giotto 7 - 20032 Cormano (MI) - oppure via FAX allo 02 66508225

(Inviando questo modulo si accettano i termini e le condizioni riportate a tergo)

Nome .... Cognome ....

Indirizzo .... n° ....

Cap .... Città .... Prov ....

Tel .... Fax .... Email ....

Ragione Sociale .... Codice fiscale ....

Partita IVA ....

DATA .... FIRMA .... ☐ Fattura

Expoelettronica - Modena

Ogni anno Expo Elettronica alla Fiera di Modena “apre” il calendario delle fiere di Elettronica & Co. L’appuntamento modenese si preannuncia ricco di espositori e prodotti, come sempre tantissimi e di vario impiego: computer, software, periferiche, telefonia fissa e mobile, videogame, home entertainment, ricezione satellitare, piccoli elettrodomestici, accessori, ricambi e strumentazioni varie. Altrettanto assortito il settore del “fai da te”, con kit per auto costruzione, surplus, schede, circuiti.
Parallelamente a Expo Elettonica si svolgono Colleziosa, mostra mercato dedicata al collezionismo in genere e Photo Cine Video per gli appassionati di macchine fotografiche e accessori nuovi, usati e da collezione.
Nell’ambito di Expo Elettronica si terrà l'evento Modena Makers, con interessanti presentazioni tecniche e dimostrative di stampanti 3D, Arduino, Droni volanti, Didattica e Open Source.

Dove: Modena
Quando: 19-20 Gennaio 2013
Orari: dalle 9.00 alle 18.00
Organizzazione: Blunautilus
info: www.expoelettronica.it

CODICE MIP 2814988

RADIANT

Dal 2002 la periodicità è divenuta quadrimestrale, con appuntamenti fissi nei mesi di gennaio, giugno e ottobre. In un’animata carrellata tra passato e futuro, la rassegna si configura in tre settori:
MOSTRA-MERCATO (apparati e componenti per telecomunicazioni, Internet e ricetrasmissioni di terra e satellitari, antenne, elettronica, informatica, telefonia statica e cellulare, hobbistica ed editoria);
BORSA-SCAMBIO (surplus radioamatoriale, informatico ed elettronico);
RADIOANTIQUARIATO (radio d’epoca, grammofoni, valvole, ricambi, dischi in vinile, schemari ed editoria specializzata).
Attualmente RADIANT, che si sviluppa su una superficie espositiva superiore ai 10.000 mq. coperti, con più di 180 espositori per edizione e oltre 40.000 visitatori annui, è considerata la più importante mostra-mercato nazionale del suo settore. Fra le categorie merceologiche ricordiamo: elettronica, informatica, telefonia, radiantismo, editoria, tv satellitare, hobbistica, surplus, radio d’epoca.

Dove: Novegro (MI)
Quando: 26-27 Gennaio 2013
Orari: dalle 9.00 alle 18.00
Organizzazione: Comis
info: www.parcoesposizioninovegro.it

CODICE MIP 2814990

Mondo Elettronica Ravenna

Nella suggestiva cornice del Palazzo Mauro de Andrè, il polo fieristico della città di Ravenna, che ospita fiere, mostre-mercato e meeting nazionali, sabato 2 e domenica 3 febbraio si svolgerà il prestigioso appuntamento dedicato al mondo dell' elettronica, dell'informatica, della telefonia, del digitale ecc.

Dove: Ravenna
Quando: 2-3 Febbraio 2013
Orari: dalle 9.00 alle 18.00
Organizzazione: Exposition Service
info: www.mondoelettronica.net

CODICE MIP 2814992

MARC di Genova

Con il Natale alle porte arriva in Fiera a Genova il 15 e il 16 dicembre il MARC, una delle migliori occasioni per lo shopping tecnologico. Attrezzature e componentistica radioamatoriale sono il nucleo storico di MARC, alle quali si affiancano e crescono informatica, telefonia, sicurezza, accessori ed editoria specializzata.

Dove: Genova
Quando: 15-16 Dicembre 2012
Orari: dalle 9.00 alle 18.00
Organizzazione: Studio Fulcro
info: www.studiofulcro.it

CODICE MIP 2797325

EXPOelettronica Vicenza

Ritorna l'appuntamento con la tappa di ExpoElettronica in Veneto, per la precisione alla Fiera di Vicenza. Le caratteristiche distintive restano quelle comuni al circuito: quantità, qualità e convenienza, per affari high tech low cost!

Dove: Vicenza
Quando: 2-3 Febbraio 2013
Orari: dalle 9.00 alle 18.00
Organizzazione: Blunautilus
info: www.expoelettronica.it

CODICE MIP 2816938

34ª Mostra regionale elettronica

Mostra/mercato regionale di elettronica, informatica, CD/DVD, telefonia. La mostra è aperta sabato fino alle 18:30, domenica fino alle ore 18.00.

Dove: Scandiano
Quando: 16-17 febbraio 2013
Orari: dalle 9.00 alle 18.30
Organizzazione: Comune di Scandiano
info: www.fierascandiano.it

CODICE MIP 2816946

Electronic Days Mantova

Informatica, computer, CD e DVD, editoria specializzata, antenne, sistemi satellitari e ricambi, batterie, apparati per radioamatori, MP3/4 e tanti altri articoli.
Parcheggio gratuito, bancomat e punto ristoro.

Dove: Mantova
Quando: 11-12 febbraio 2013
Orari: dalle 9.00 alle 18.00
Organizzazione: Italfiere
info: www.italfiere.net

CODICE MIP 2798102

DI GIUSEPPE LA ROSA

CONTROLLO LUCI MUSICALI *tramite porta USB*

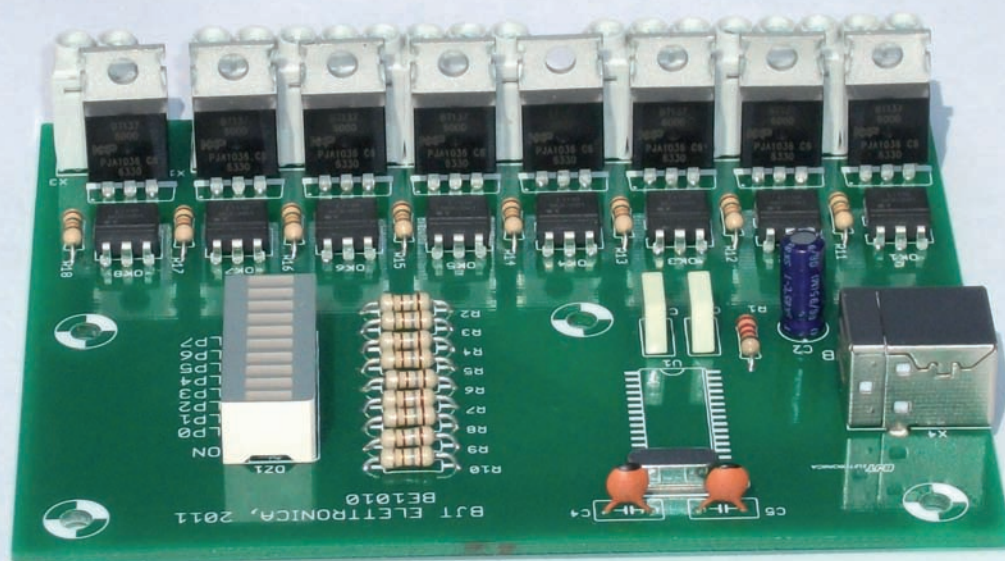



Figura 1: il progetto finito.

Costruiamo un completo controllo luci a 8 canali tramite porta USB, con la possibilità di pilotare le lampade a tempo della musica riprodotta dal PC.

Se siete in possesso di un qualunque PC dotato di sistema operativo Windows (a 32 bit) con porta USB, allora non privatevi della possibilità di costruire una scheda che vi permetterà di creare un'infinità di giochi di luci personalizzati.
La scheda, già alimentata direttamente da 230 V, comprende: LED monitor dei giochi, uscite di potenza da 300 W per canale e isolamento optoelettronico; il quale permette di proteggere adeguatamente il computer.
Un facile programma in grado di gestire i giochi anche al ritmo della musica riprodotta dal PC stesso, rende tutto, oltre che divertente, estremamente facile e professionale.

Figura 2: schema elettrico della scheda USB.

SCHEMA ELETTRICO

Lo schema elettrico visibile in **figura 2** è basato sul circuito integrato U1, un CH341A usato dall'industria per realizzare convertitori USB seriali e USB paralleli.
Il chip CH341A (U1) è configurabile in tre modalità:

• UART - convertitore USB a porta seriale (livelli logici TTL);
• PRINT - convertitore USB a porta parallela;
• EPP / MEM / I²C / Digital I/O - uscite e ingressi e BUS I²C supportati da chiamate API;

In questo progetto abbiamo usato la terza modalità, cioè l'EPP / MEM / I²C / Digital I/O.
Questa modalità può essere configurata collegando il pin 1 (U1) e la massa a una resistenza di 2,2 k (R1).
L'uso di questa configurazione comporta il caricamento di un driver di Windows: accessibile da diversi linguaggi di programmazione tra cui ProfiLab-Expert 4.0 utilizzato, nel nostro caso, attraverso un'API-DLL (CH341API.DLL).
I pin che vanno dal piedino 15 al 22 del CH341A (U1) sono impostati come 8 uscite che possono essere scritte tramite chiamate API.

Per chi volesse approfondire le altre modalità UART, PRINT, e la modalità usata in questo progetto può consultare il Datasheet.

Come si può vedere dallo schema elettrico, dai piedini di U1 (piedini da 15 a 22) giungono i livelli logici 0 e 1. Questi alimentano i LED della bargraph DZ1 i quali, posti in serie ai fotoaccoppiatori OK1 a OK8, si portano in conduzione; i fotodiodi contenuti al loro interno eccitano il GATE dei TRIAC T1 a T8, causando l'accensione delle lampade collegate alle morsettiere X1 e X2.
I fotoaccoppiatori offrono un isolamento di circa 4000 V in modo che il PC risulta perfettamente isolato dalla tensione di rete.
La tensione di rete viene prelevata dal morsetto X3 ed è usata solo per alimentare le lampade a 230 V.
L'alimentazione a 5 V occorrente al funzionamento di U1 e dei fotoaccoppiatori viene presa direttamente dalla porta USB connettore X4.
I due LED DZ1-1 e DZ1-2 della bargraph DZ1 sono connessi in serie e, collegati a 5 V, hanno la funzione di monitorare la connessione alla porta USB. Il quarzo Q1 fornisce la frequenza di clock di 12 MHz necessaria al funzionamento del circuito integrato U1.

REALIZZAZIONE PRATICA

Per la concretizzazione di questo progetto **(figura 1)**, si deve utilizzare il master di **figura 4**, per la realizzazione del **PCB** e i componenti elencati nella **lista componenti**.
Una volta realizzato il circuito stampato potete iniziare a montare tutti i componenti, disponendoli come in **figura 3**.
Si raccomanda di montare per primo il circuito integrato in SMD U1 dal lato saldature, rispettando l'orientamento della tacca sullo stampato **(figura 5)** utilizzando l'apposita pasta saldante e un saldatore ad aria calda.
Dopo aver saldato U1 e aver controllato le saldatura e la correttezza dei piedini si può passare a montare tutti gli elementi a basso profilo sul lato componenti: resistenze, condensatori, la Bargraph DZ1 e i fotoaccoppiatori. Si può passare infine ai TRIAC, al connettore USB X4 e ai morsetti.
Finito l'assemblaggio della scheda passate

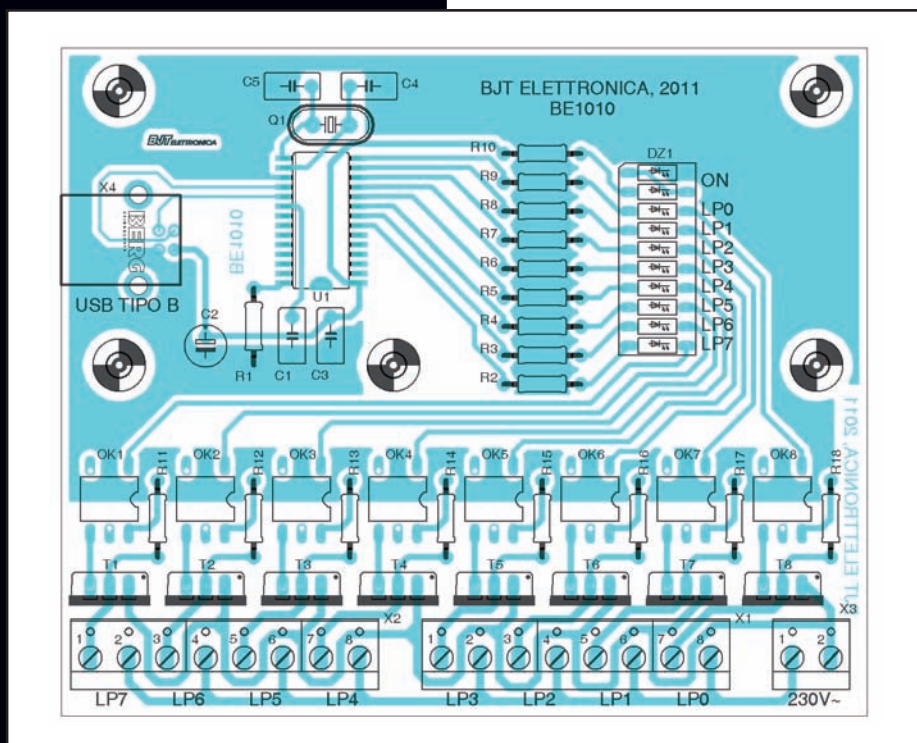



Figura 3: piano di montaggio.

Figura 5: circuito stampato vista lato saldature.

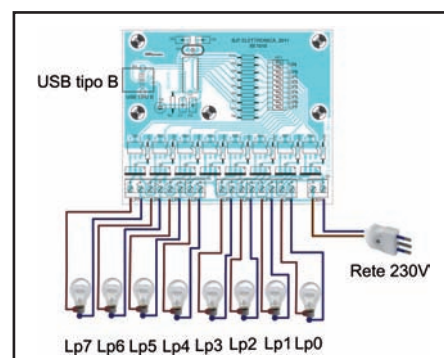



Figura 6: schema di collegamento alla rete e alle lampade.

alla verifica del montaggio. Dopo che si è verificato che tutto sia saldato correttamente si può passare al collaudo. Per fare ciò rinvio alla lettura dei prossimi paragrafi.

INSTALLAZIONE DEL SOFTWARE

Per installare il programma è necessario che prima lo scarichiate dal sito della rivista.

È importante che prima di eseguire l'installazione non venga collegata la scheda al PC. Finita l'installazione del software è possibile collegare la scheda al PC; si avvierà l'installazione plug & play di Windows, basta digitare il seguente percorso: C:\Programmi\Control light music 1.0\DRIVER per installare il driver della scheda. Con questa operazione si conclude la fase di installazione, il che permette di passare al suo utilizzo.

Figura 4: circuito stampato in scala 1:1 (lato saldature).

COLLEGAMENTO ALLA RETE

Attenzione: Il dispositivo lavora alla tensione di rete, per cui evitare di toccare le parti metalliche, è consigliabile installarlo in un contenitore di plastica. Per il collegamento delle lampade bisogna realizzare il collegamento della **figura 6**. La massima potenza applicabile per canale è di 300 W.

I cavi per il collegamento delle lampade devono avere una sezione minima di 1,5 mm². Il cavo per collegamento al PC deve essere USB tipo B (cavo USB utilizzato per la stampante) massima lunghezza 2 metri.

DESCRIZIONE DEL FUNZIONAMENTO DEL SOFTWARE

In **figura 7** è visibile la schermata del software in cui sono numerate tutte le funzioni dei pulsanti:

1. Indicatore analogico di livello musicale;
2. monitor dei giochi luminosi;
3. selettore per la scelta di esecuzione a tempo di Musica o Velocità (da 0 a 30);
4. selettore per la scelta di esecuzione Automatica (tempo di Musica o Velocità) o Manuale (azionamento tramite interruttori e pulsanti vedi punto 10);
5. selettore per la scelta di arresto o avvio dei giochi di luce: in posizione Start avvio giochi di luce, in posizione Stop arresto giochi di luce;
6. pulsate che esegue il Reset delle sequenze;

Figura 7: schermata principale del software di controllo.

LISTA COMPONENTI

Sigla	Valore
C1	10 nF poliestere
C2	22 µF 25 V elettrolitico
C3	100 nF poliestere
C4÷C5	15 pF ceramico
R1	2,2 kΩ 1/4 W
R2÷R10	180 Ω 1/4 W
R11÷R18	15 k Ω 1/4 W
DZ1	Bargraph verde
OK1÷OK8	MOC3021
Q1	12 MHz quarzo
T1÷T8	BT137 triac
U1	CH341A
X1÷X2	Morsetti a 8 poli
X3	Morsetti a 2 poli
X4	USB tipo B

7. input per l'inserimento del valore numerico della sequenza in cui va inseguito il Loop.
8. Indicatore delle sequenze eseguite.
9. Cursore della velocità di scansione delle sequenze da 0 a 30
10. Gruppo di 8 pulsanti per comando monostabile e 8 interruttori per il comando bistabile più due pulsanti che mettono le lampade o tutte ON o Tutte OFF.

Prima di procedere all'utilizzo **(figura 8)** è necessario selezionare la scheda audio, cliccare sull'icona cacciavite come mostra la **figura 8**. Si aprirà il menu Configuration, cliccare su Scheda Audio (AUD1), selezionare la scheda audio, cliccaee OK e poi Close. Quindi occorre chiudere il programma e riaprirlo. In questo modo è stata selezionata la scheda audio che il programma utilizzerà per eseguire giochi luminosi a tempo musicale. Per creare un gioco di luce **(figura 9)**, cliccare sull'icona Cacciavite, si aprirà il menu Configuration. Cliccare su ROM (ROM1), si aprirà una tabella da cui spuntare Binary. Basta fare doppio clic su una colonna e lo zero diventerà uno e viceversa.
Impostate cosi la sequenza desiderata; contate il numero di righe create, cliccate su Save.
Load permette di richiamare i file già salvati in precedenza contenenti i giochi di luce. Cliccate su OK, poi su Close.
Inserite il valore del numero di righe realizzate in Loop e premete invio. Chiudete il programma e riapritelo.

A questo punto il vostro gioco di luce può essere eseguito nelle due modalità a tempo di Musica o Velocità regolabile da cursore.

CONCLUSIONE

Come già è stato detto in precedenza, il software è stato sviluppato con Profi-Lab-Expert 4.0. Il programma ProfiLab-Expert 4.0 permette di sviluppare progetti di misura analogica e digitale basati su PC. Per lo sviluppo non occorre essere programmatori esperti, è possibile creare i progetti eseguendo uno schema elettrico. Basta aggiungere pulsanti, display ecc. allo schema e costruire il progetto passo dopo passo. Il software sorgente e tutti gli altri file necessari alla realizzazione o personalizzazione di questo progetto sono disponibili per il download dal sito di Fare Elettronica. Al seguente link potete vedere la scheda in funzione:
http://www.youtube.com/watch?feature=player_embedded&v=VyMslzAfPiA ❑

CODICE MIP 2831859

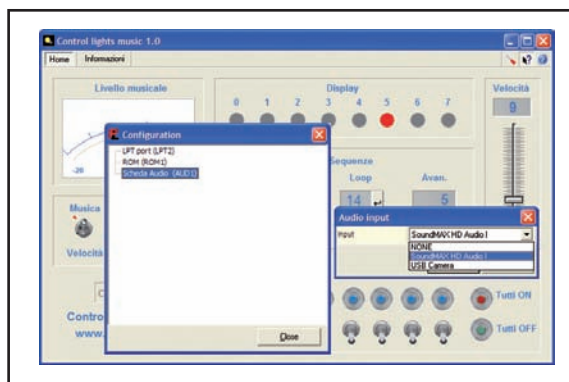



Figura 8: schermate per la selezione della scheda audio.

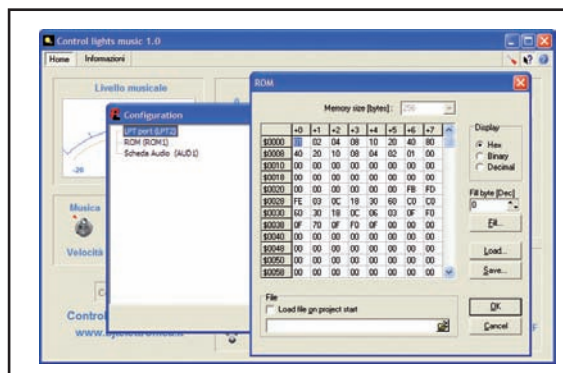



Figura 9: schermata per la creazione di sequenze luminose.

di MAURIZIO GIOVANARDI

Regolatore per giostra DA CACCIA

Con questo semplice progetto possiamo realizzare un regolatore di velocità che può essere utile in molte applicazioni: giostre per cacciatori, ventole, motori in corrente continua o dimmer per lampade a 12-24 V. Il carico massimo applicabile è di circa 4 A, valore limite oltre il quale scatta una protezione da sovracorrente.

Questo circuito nasce grazie a un amico cacciatore che ogni settimana si presentava a casa mia per farsi riparare il regolatore di velocità della sua giostra per la caccia, un apparecchio che si usa per attirare i volatili. Aveva provato tanti regolatori, ma tutti "lineari". Questi dispositivi, quasi sempre autocostruiti, altro non erano che regolatori di tensione lineari che spesso si bruciavano per l'eccessivo calore generato (piccolissime alette di raffreddamento montate dentro a piccoli contenitori ermetici) oppure rimanevano "vittime" dei cacciatori stessi che pretendevano di collegare a un unico regolatore tre o più motori. Per risolvere il problema del mio amico cacciatore ho pensato di realizzare un regolatore di tensione PWM (modulazione a larghezza di impulsi) che ha il pregio di essere compatto, dissipa pochissima potenza in calore (aumentando l'autonomia della batteria) e in più ha una protezione da sovracorrente.

IL CIRCUITO

Il circuito ruota attorno al microcontrollore ATtiny24A di ATMEL (U1) che, come si può vedere dallo schema elettrico riportato in **figura 1**, svolge tutte le funzioni richieste.
A ogni pressione del pulsante collegato alle piazzole X1 e X2 il micro incrementa del 20% il duty-cicle del PWM il quale co-

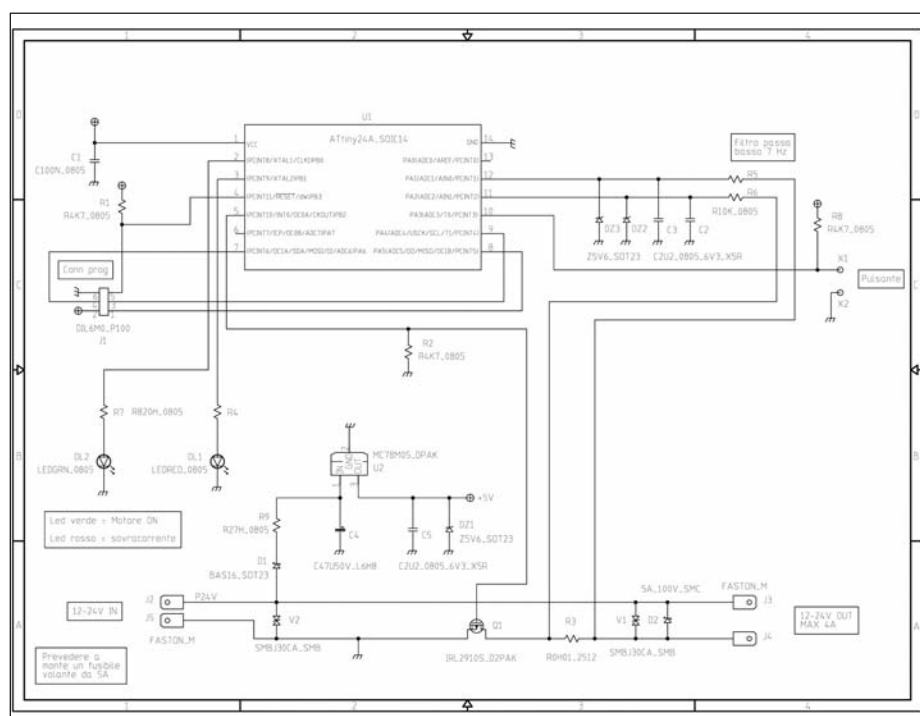



Figura 1: schema elettrico.

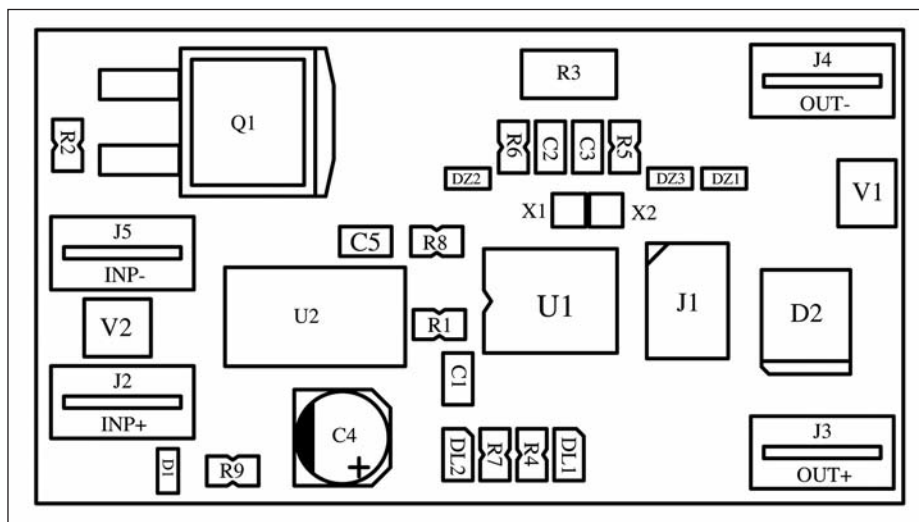



Figura 2: piano di assemblaggio.

manda il MOSFET Q1, fino ad arrivare al 100% che corrisponde a Q1 sempre acceso.
L'Attiny24A lavora con un clock di 8 MHz sfruttando l'oscillatore interno e genera un PWM con frequenza di 15 kHz, appena al di sopra della soglia udibile dall'orecchio umano.
Nello schema sono presenti due diodi LED di segnalazione: quello verde (DL2) si accende quando al carico arriva tensione, mentre quello rosso (DL1) si accende se la corrente erogata dalla scheda supera 4 A.
Come funziona la protezione da sovracorrente? A valle di Q1 c'è una piccola resistenza di shunt da 10 mohm (R3) che, quando è attraversata dalla corrente del carico, genera una differenza di potenziale ai suoi capi.
Questa d.d.p. viene filtrata dal filtro passa basso C2/R6 e C3/R5 (evita che gli spunti di corrente possano sembrare sovracorrenti) e va a finire nel convertitore analogico / digitale di U1 (pin 11 e 12).
Si tratta di un A/D converter a 10 bit differenziale, con guadagno pari a 20, molto comodo da utilizzare essendo integrato all'interno del microcontrollore.
Il firmware controlla continuamente il valore letto dal convertitore A/D e, non appena si supera la soglia impostata, vengono spenti Q1 e il LED verde, mentre viene acceso il LED rosso.
In questa situazione la scheda è in blocco e per farla ripartire occorre disalimentarla temporaneamente.
Voglio sottolineare che la protezione da sovracorrente non sostituisce il fusibile che si mette di solito sul positivo di alimentazione di qualsiasi circuito. Consiglio di impiegarne uno da 5 A volante per maggior sicurezza.
Lo stadio di alimentazione a 5 V dell'Attiny24A è veramente semplice, si tratta di un regolatore lineare (U2) che permette di alimentare la scheda con tensioni comprese tra 9 e 24 V.
Da notare che la resistenza R9 da 27 ohm ha il compito di proteggere il diodo di anti-inversione D1 all'accensione, quando C4 praticamente gli presenta un cortocircuito.
I TRANSORB V1 e V2 proteggono la scheda dalle scariche elettrostatiche, mentre D2 (diodo di ricircolo) taglia le extratensioni generate dal carico induttivo negli istanti in cui si spegne Q1.

IL FIRMWARE

Il firmware, come potete vedere dal listato, è stato scritto in ASSEMBLER, soluzione comoda grazie alla ricchezza del linguaggio di programmazione dell'ATMEL e alla semplicità delle funzioni richieste in questo progetto.
Nelle prime righe del listato vengono definite le costanti, tra cui "SOGLIA_CORRENTE" che corrisponde al famoso valore di 4 A da monitorare.
Facciamo un esempio: nel caso volessimo portare questa soglia a 3 A, quale valore assumerà SOGLIA_CORRENTE?
Una corrente di 3 A sul carico genera ai capi di R3 una d.d.p. di 30 mV che il micro moltiplica internamente per 20, quindi al convertitore A/D arrivano 0,6 V.
Questo convertitore ha una risoluzione di 10 bit e lavora con un riferimento interno da 1,1 V, quindi ogni bit equivale a 1,1 V / 1024 = 1,074 mV.
Infine 0,6 V equivalgono a 0,6V / 0,001074 V = 558. La lettura A/D da monitorare è 558 decimale che corrisponde a 1000101110 binario.

LISTA COMPONENTI

Sigla	Valore
R1	4,7 Kohm 125 mW SMD 0805
R2	4,7 Kohm 125 mW SMD 0805
R3	0,01 ohm 1 W SMD 2512
R4	820 ohm 125 mW SMD 0805
R5	10 Kohm 125 mW SMD 0805
R6	10 Kohm 125 mW SMD 0805
R7	820 ohm 125 mW SMD 0805
R8	4,7 Kohm 125 mW SMD 0805
R9	27 ohm 125 mW SMD 0805
C1	100 nF 25 V SMD 0805
C2	2,2 uF 6,3 V SMD 0805
C3	2,2 uF 6,3 V SMD 0805
C4	47 uF 50 V SMD 6X8
C5	2,2 uF 6,3 V SMD 0805
U1	ATMEL ATtiny24A SMD SOIC14
U2	MC78M05 SMD DPAK
D1	BAS16 SMD SOT-23
D2	S5B SMD SMC
J1	Pettine maschio 2X3 pin passo 2,54 mm
J2	Faston maschio da c.s.
J3	Faston maschio da c.s.
J4	Faston maschio da c.s.
J5	Faston maschio da c.s.
Q1	IRF IRL2910S SMD D2PAK
DL1	Led rosso SMD 0805
DL2	Led verde SMD 0805
V1	SMBJ30CA SMD SMB
V2	SMBJ30CA SMD SMB
DZ1	ZENER 5,6 V 200 mW SMD SOT-23
DZ2	ZENER 5,6 V 200 mW SMD SOT-23
DZ3	ZENER 5,6 V 200 mW SMD SOT-23

Figura 3: traccia lato componenti.

LISTATO 1

```
Listato 1
;*****************************************************************************
;*
;* Scheda: Regolatore PWM per giostra-richiamo
;*
;* Versione: 1
;*
;* Target: ATtiny24A con clock a 8MHz > 0.125us per ciclo
;*
;* Data: ottobre 2012
;*
;* Descrizione: Spingendo il pulsante regolo il PWM al 0% 20% 40%
;*              60% 80% 100%. Per evitare rimbalzi ogni volta che
;*                      premo il pulsante lo inibisco per circa 1 secondo.
;*                      Il led verde significa motore on mentre quello
;*              rosso segnala l'intervento della protezione da
;*                      sovracorrente (con il convertitore A/D controllo
;*                      la corrente erogata, se supera i 4A viene impostato
;*                      il PWM a 0 ed occorre spegnere e riaccendere la
;*                      scheda per farla ripartire).
;*
;*
;* PWM 15 KHz = 65us / 5 = 13us
;*
;****************************************************************************

.include "tn24Adef.inc"

.def temp=r20                               ;registro temporaneo
.def dutycicle=r21
.def temp2=r22

.equ LED_VERDE=PB0
.equ LED_ROSSO=PB1
.equ COMANDO_MOS=PB2
.equ PULSANTE=PA3
.equ SOGLIA_CORRENTE=0b10111010             ;soglia pari a 4A

;*** Codice del programma ***

.cseg

.org$0000

rjmp init                    ;Reset handler

;*** Inizializzazione generale ***

init:      lditemp,low(RAMEND)      ;Settaggio stack negli indirizzi alti della RAM
     out   SPL,temp
     ldi   temp,0b11110001          ;porta A tutte uscite tranne PA3 PA2 PA1
     out   DDRA,temp
     clr   temp                     ; out A a 0 ed inp senza pullup
     out   PORTA,temp
     ldi   temp,0b11110111          ;porta B tutte uscite tranne PB3
     out   DDRB,temp
     clr   temp                     ;out B a 0 ed inp senza pullup
     out   PORTB,temp
     ldi   temp,0b00100111
     out   ADCSRA,temp              ;adc con prescaler 128 in free run
     ldi   temp,0b00010000
     out   ADCSRB,temp              ;adlar=1 perdo bit0 e bit1 meno significativi
     ldi   temp,0b10001101          ;ADC con ref int (1,1V) e ADC2 inp differenz gain 20
     out   ADMUX,temp               ;Fondoscala ADC=5,5A
     ldi   temp,0b10100000
     out   ADCSRA,temp              ;abilito adc
```

LISTATO 1 segue

```
      ldi   temp,0b11100000          ;avvio ADC
      out   ADCSRA,temp
      ldi   temp,0b00101001          ;abilito watchdog con timeout 8s
      out   WDTCSR,temp
      wdr                            ;watchdog reset
      cli                            ;disabilita interrupt
      clr   dutycicle                ;fermo il motore

      rcall ritardo_init             ;ritardo iniziale per far partire l'ADC

      clr   temp
      out   TIMSK1,temp              ;disabilito interrupt
      out   TCCR1B,temp              ;fermo il contatore
      out   TCCR1A,temp              ;conteggio normale
      ldi   temp,0b10000101          ;carico 34286 come valore iniziale
      out   TCNT1H,temp
      ldi   temp,0b11101110
      out   TCNT1L,temp
      ldi   temp,0b00000100          ;faccio partire il contatore con prescaler 256
      out   TCCR1B,temp

inizio:        sbic   PINA,PULSANTE
      rjmp  frullo
      clr   temp
      out   TIMSK1,temp              ;disabilito interrupt
      out   TCCR1B,temp              ;fermo il contatore
      ldi   temp,0b00000001
```

Scarica tutto il listato completo su www.farelettronica.com

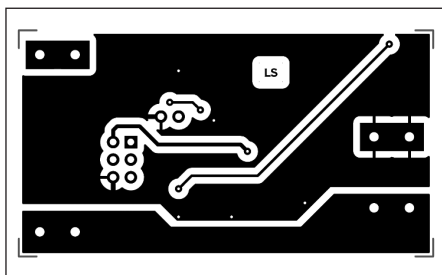



Figura 4: traccia lato saldature.

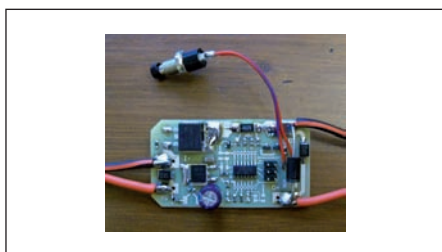



Figura 5: prototipo iniziale.

Per comodità, visto che i registri interni del microcontrollore sono a 8 bit e non è richiesta una grande precisione su questa lettura, non consideriamo i due bit meno significativi e quindi otteniamo SOGLIA_CORRENTE = 10001011.

Dopo la definizione delle costanti, vengono inizializzate alcune periferiche interne. Il converitore A/D, per esempio, viene settato nella modalità FREE RUNNING (funzionamento continuo) e viene richiamata la routine ritardo_init per evitare di leggere il suo primo valore campionato che non è significativo. A questo punto entriamo in un loop diviso in cinque parti (una per ogni incremento del 20% di duty-cicle) in cui viene letto il pulsante di comando e aggiornato di conseguenza il valore di un registro chiamato "dutycicle", registro che può assumere valori da 0 (dutycicle 0%) a 5 (dutycicle 100%).
In ognuna delle cinque sezioni, in base al valore assunto da questo registro, il comando del gate di Q1 viene dato o tolto per rispettare il dutycicle impostato.
Da notare che, tramite un contatore interno a 16 bit, dopo ogni pressione del pulsante la sua lettura viene inibita per circa un secondo al fine di evitare rimbalzi del pulsante stesso.
All'interno del loop viene monitorata anche un'eventuale sovracorrente. Se viene rilevata si entra in un ciclo infinito.

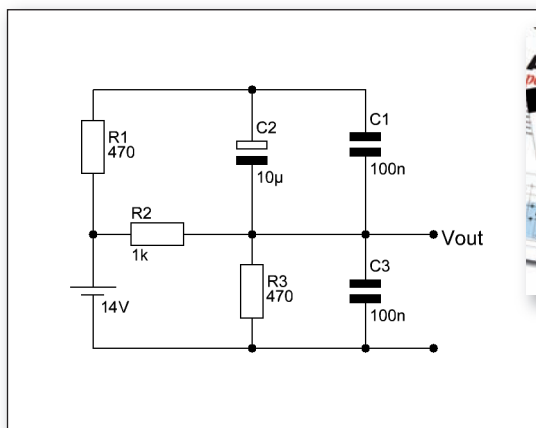
Il file .hex necessario per poter programmare il microcontrollore è scaricabile dal sito Internet di *Fare Elettronica*; se avete necessità di modificare il codice sorgente e ricompilare il file .hex potete utilizzare l'ambiente di sviluppo integrato "AVR Studio" che potete scaricare gratuitamente dal sito internet www.atmel.com.

REALIZZAZIONE PRATICA

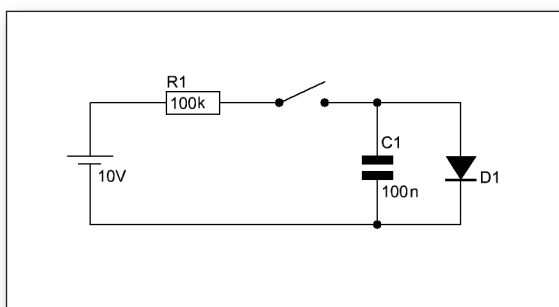
Questo progetto è stato realizzato utilizzando componenti SMD montati su circuito stampato a doppia faccia, scelta che ha permesso di renderlo veramente compatto come dimensioni; in figura 5 potete vedere il prototipo iniziale.
Se volete potete scaricare la traccia lato componenti e lato saldature dal sito Internet, oppure, per avere una più facile reperibilità dei componenti, potete utilizzare componenti discreti montati su basetta millefori, a voi la scelta.
In ogni caso ponete attenzione a orientare i componenti con polarità, seguendo il piano di assemblaggio di **figura 2**. ❑

CODICE MIP 2831453

330

Le risposte ai quiz “facile” e “difficile” vanno inviate esclusivamente compilando il modulo su www.farelettronica.com/eq specificando la parola chiave ***“Miller”****.*
Le risposte e i vincitori (previa autorizzazione) sono pubblicati alla pagina www.farelettronica.com/eq a partire dal 15 del mese successivo alla pubblicazione sulla rivista.
A tutti i partecipanti verrà assegnato un buono sconto del 10% (validità 3 mesi dalla data di assegnazione) utilizzabile per un prossimo acquisto su www.ieshop.it

Abbonamento per un anno al club di Fare Elettronica

facile

Nel circuito di figura, supponendo nulle le perdite nei condensatori, si determini la tensione di uscita Vout.

difficile

Considerando per il diodo D1 una tensione di soglia di 0,7 V, determinare il tempo impiegato dal condensatore per raggiungere il massimo valore di tensione ai suoi capi dopo la chiusura dell’interruttore.

Abbonamento per un anno a Fare Elettronica

di GIOVANNI DI MARIA

PWM avanzato *(parte seconda)* CON IL PIC

In questa seconda parte approfondiamo le conoscenze e le tecniche legate all'utilizzo e alla gestione del PWM di tipo hardware.

IL PWM VIA HARDWARE

Trattiamo adesso le tecniche di generazione del PWM attraverso l'hardware dedicato, presente all'interno dei microcontrollori PIC. I modelli base di MCU sono sprovvisti di tale possibilità.
Il modulo che permette tale possibilità è chiamato modulo CCP (Capture/Compare/PWM) ed è predisposto anche per altre funzionalità. Il mikroBasic lo gestisce con una semplicità estrema, semplificando al massimo il lavoro del progettista e del programmatore.

Le procedure del mikroBasic

Il mikroBasic dispone di quattro procedure (infatti non ritornano alcun valore), semplicissime, per gestire totalmente il segnale

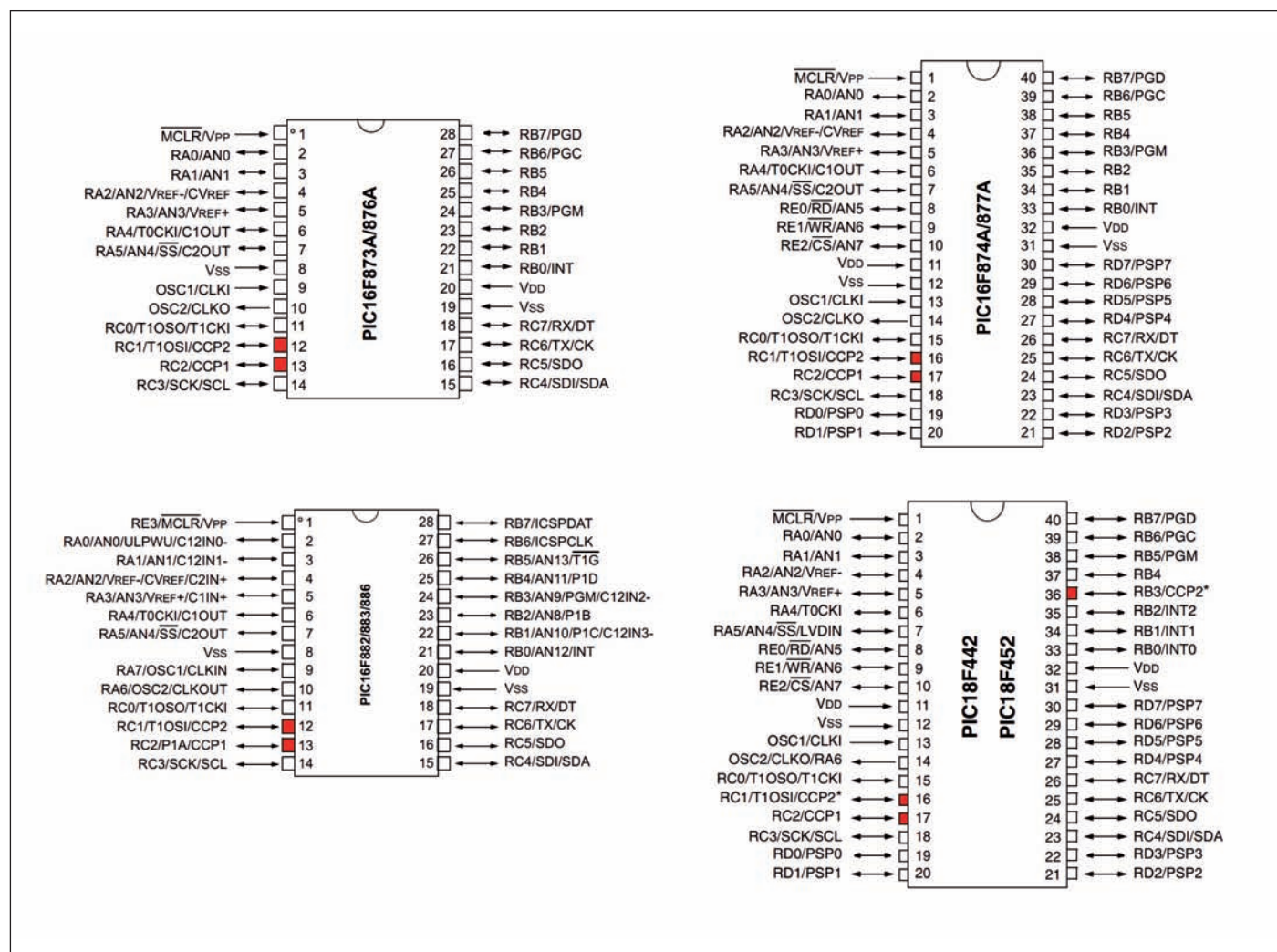



Figura 1: i moduli CCP in alcuni modelli di PIC.

PWM. Esse sono le seguenti:

- PWM1_Init
- PWM1_Set_Duty
- PWM1_Start
- PWM1_Stop

Come si nota, tali procedure iniziano tutte con il suffisso "PWM" seguito da un numero. Quest'ultimo rappresenta il canale PWM utilizzato. Ci sono modelli di microcontrollore che dispongono di uno solo, due o più canali. Occorre semplicemente sostituire tale numero con quello del canale utilizzato (es: PWM2_init oppure PWM3_start).

PWM1_Init

È una procedura che inizializza il modulo PWM, impostandolo alla frequenza specificata, con un duty cycle pari allo 0%. Il parametro è espresso in Hz. Essa deve essere invocata sempre prima di utilizzare le altre procedure.

Se occorre utilizzare la seconda porta PWM, basta invocare la procedura PWM2_Init.

Sintassi:

```
PWM1_Init(const freq as longint)
```

Esempio:

```
PWM1_Init(3000)
```

PWM1_Set_Duty

Questa procedura imposta il tasso del duty cycle, non espresso in percentuale ma in valore di tipo byte (a 8 bit), da 0 a 255. Pertanto specificando 0 si imposta una percentuale di ciclo attivo pari allo 0%, specificando 127 si imposta al 50%, specificando 255 si imposta al 100%. Valori intermedi selezionano percentuali intermedie. Il calcolo è possibile grazie ad una semplice proporzione.

La **tabella 1** sottostante mostra le principali percentuali accanto ai valori parametrici da utilizzare.

LISTATO 1

```
rem **************************
rem *       LISTATO 1        *
rem *     PWM Hardware       *
rem *       10% e 60%        *
rem *  by Giovanni Di Maria  *
rem **************************
program pwm
main:
     rem —-Porte in Output—-
     trisc.1=0
     trisc.2=0
     rem —-Azzera porte——
     portc.1=0
     portc.2=0
     rem —- Inizializza CCP1 e CCP2—-
     PWM1_Init(5000)
     PWM2_Init(5000)
     rem —-Duty cycle al 10% e 60%—
     PWM1_Set_Duty(25)
     PWM2_Set_Duty(153)
     rem—-Iniziano i 2 PWM——
     PWM1_Start()
     PWM2_Start()
     rem —- Dopo 10 sec. il primo termina—-
     delay_ms(10000)
     PWM1_Stop()
     rem —- Dopo altri 10 sec. il secondo termina—-
     delay_ms(10000)
     PWM2_Stop()
end.
```

Sintassi:

```
PWM1_Set_Duty(dim duty_ratio as byte)
```

Esempio:

```
PWM1_Set_Duty(192)    '——Imposta al 75%
```

PWM1_Start

Fa partire la generazione del segnale PWM. Questa procedura non è "bloccante", nel senso che non congela l'esecuzione del programma, come farebbe, ad esempio, una temporizzazione (delay_ms(10000)). Una volta che la procedura è richiamata, il normale flusso del programma continua regolarmente dalla successiva istruzione. L'erogazione del segnale PWM è affidata, in background, dall'hardware del PIC.

Sintassi:

```
PWM1_Start()
```

Esempio:

```
PWM1_Start()
```

PWM1_Stop

Interrompe immediatamente la generazione del segnale PWM.

Sintassi:

```
PWM1_Stop()
```

Esempio:

```
PWM1_Stop()
```

IL PWM NEI VARI MODELLI DI PIC

Come detto prima, non tutti i modelli sono predisposti per contenere il modulo CCP.

Schema elettrico

Lo schema elettrico per sperimentare con il PWM hardware è minimale: due diodi LED collegati, stavolta, alle porte RC1 e RC2 del PIC 16F877A. Le due porte corrispondono ai moduli CCP2 e CCP1.

Un segnale PWM al 10% e uno al 60%

Bene, proviamo subito a generare due segnali PWM: il primo con duty cycle al 10%, quindi con potenza reale molto bassa; il secondo con duty cycle del 60%.

Entrambi i segnali iniziano nello stesso istante, ma il primo dura 10 secondi mentre il secondo 20 secondi.

La frequenza dei due segnali deve es-

TABELLA 1:

Percentuale	Valore
Percentuale: 0%	Valore: 0
Percentuale: 10%	Valore: 25
Percentuale: 20%	Valore: 51
Percentuale: 30%	Valore: 76
Percentuale: 40%	Valore: 102
Percentuale: 50%	Valore: 127
Percentuale: 60%	Valore: 153
Percentuale: 70%	Valore: 178
Percentuale: 80%	Valore: 204
Percentuale: 90%	Valore: 229
Percentuale: 100%	Valore: 255

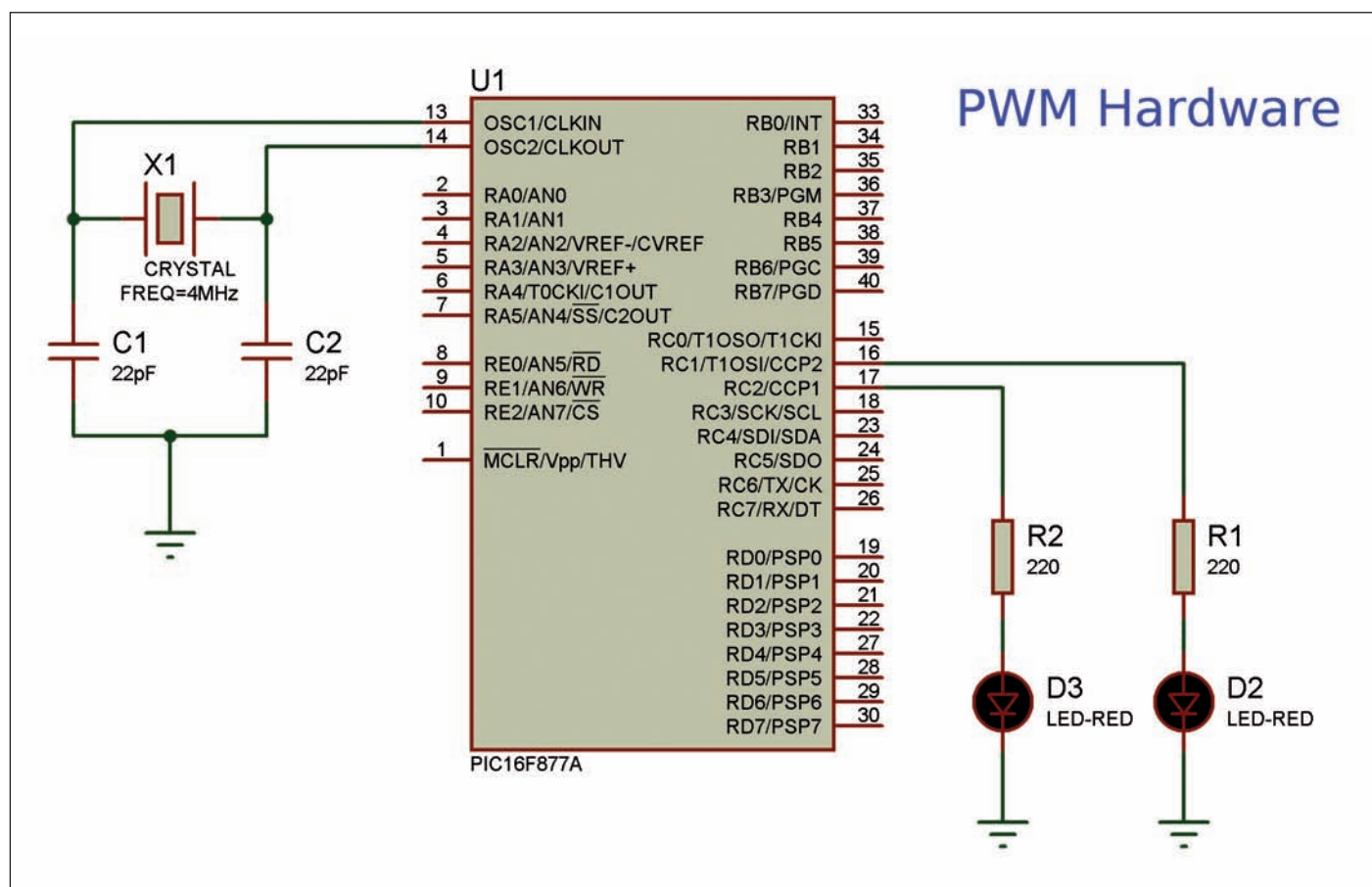



Figura 2: schema elettrico per la sperimentazione del PWM hardware.

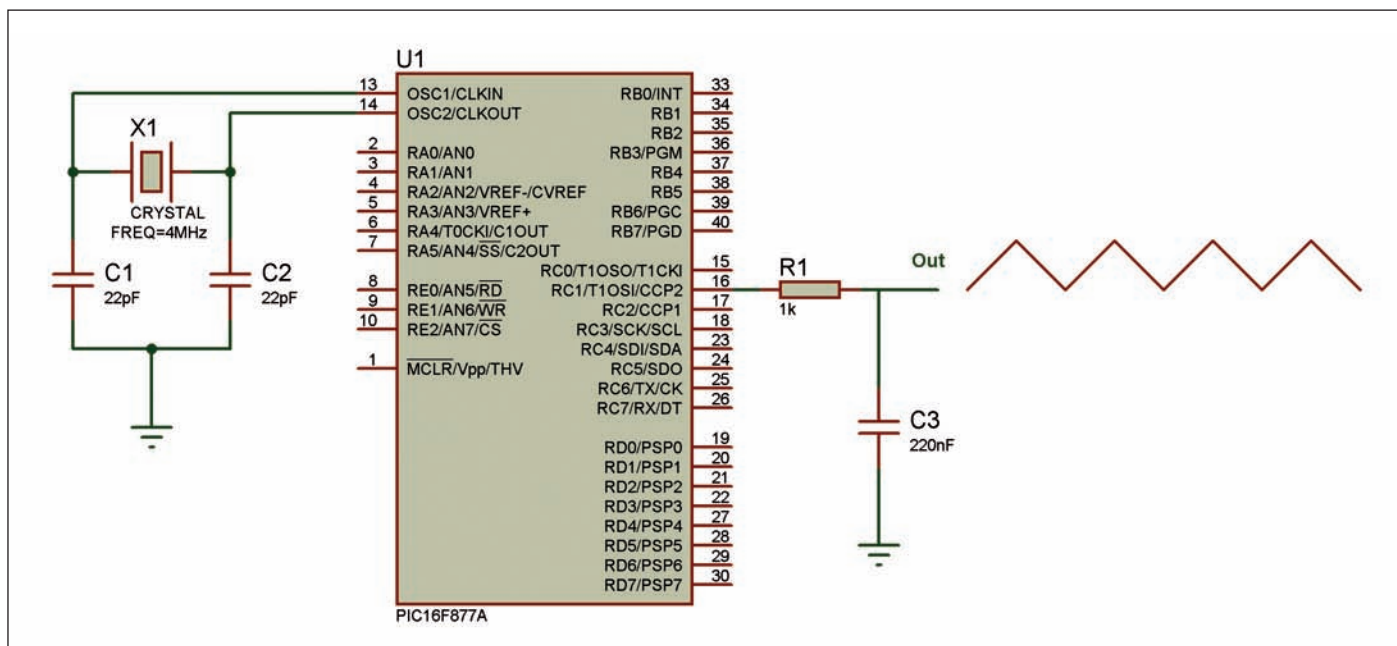



Figura 3: schema elettrico del generatore di onda triangolare.

sere la medesima (noi la fissiamo a 5 Khz), poiché il PIC, per la loro riproduzione, utilizza il Timer2.
Il **listato 1** è di una semplicità disarmante. Dopo l'inizializzazione dei moduli CCP1 e CCP2 (a 5 Khz) e settaggio del loro duty cycle (10% e 60%) i due segnali iniziano subito.
Trascorsi i primi 10 secondi, il primo segnale termina. Trascorsi ulteriori 10 secondi, anche il secondo segnale conclude il suo ciclo.
Teniamo a far notare che il PIC, durante la generazione del segnale PWM, non è "congelato" ma può svolgere liberamente qualsiasi operazione.
Ad esempio, nel frangente, si può fare lampeggiare un altro diodo LED, a bassa frequenza. Viceversa, la procedura "delay_ms" pone in condizione di attesa il microcontrollore, bloccandolo di fatto.

UN SEMPLICE GENERATORE DI ONDA TRIANGOLARE

L'utilizzo del PWM hardware permette di realizzare facilmente progetti altrimenti di complicata attuazione. Il prossimo esperimento prevede la costruzione di un semplice generatore di onda triangolare.
Il segnale PWM ha il compito, in questo caso, di produrre un treno di impulsi che, con la intrinseca ampiezza variabile, simuli la tensione ai vari gradini di salita e di discesa. L'illustrazione chiarisce meglio l'idea **(figura 3)**. Il segnale triangolare, nel nostro esempio, è composto da 20 campioni, di cui 11 tensioni in salita e 9 in discesa.
I campioni vengono riprodotti continuamente, in un ciclo infinito.

Schema elettrico

Lo schema elettrico è semplicissimo. Si tratta unicamente di un microcontrollore. L'uscita dell'onda triangolare è localizzata sulla porta RC1, corrispondente al modulo CCP2, sulla quale è presente un

LISTATO 2

```
rem **************************
rem *       LISTATO 2        *
rem *     PWM Hardware       *
rem *   Onda triangolare     *
rem *  by Giovanni Di Maria   *
rem **************************
program pwm
dim k as word
main:
     trisc.1=0          'Porta in Output
     portc.1=0          'Azzera porta
     PWM2_Init(10000)   'Inizializza CCP2
     PWM2_Start()
     while true
          rem ——Onda triangolare in SALITA—-
           for k=0 to 255 step 25
               PWM2_Set_Duty(k)   'Cambia duty
               delay_us(500)
           next k
           rem ——Onda triangolare in DISCESA—-
           for k=225 to 25 step -25
               PWM2_Set_Duty(k)   'Cambia duty
               delay_us(500)
           next k
     wend
end.
```

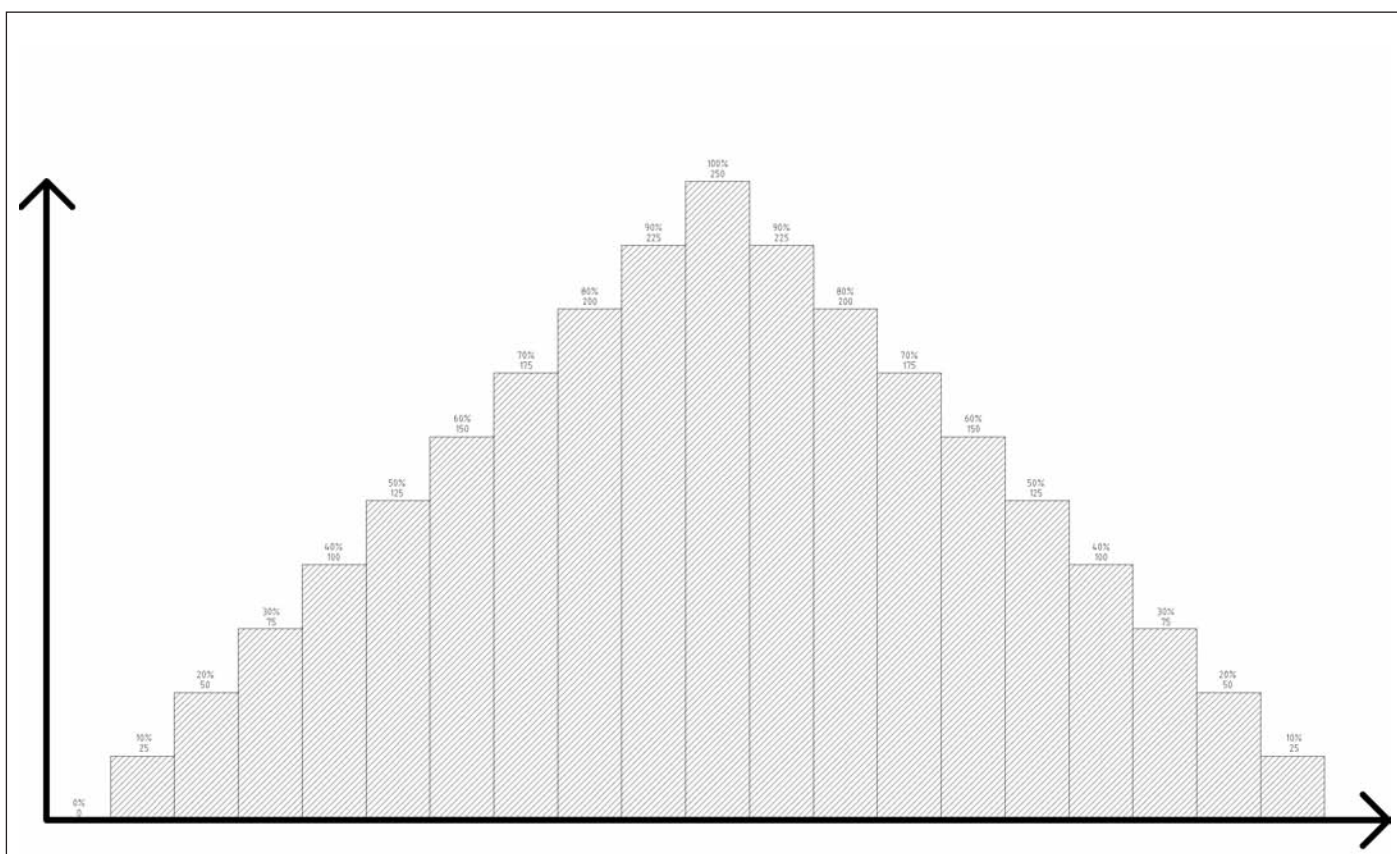
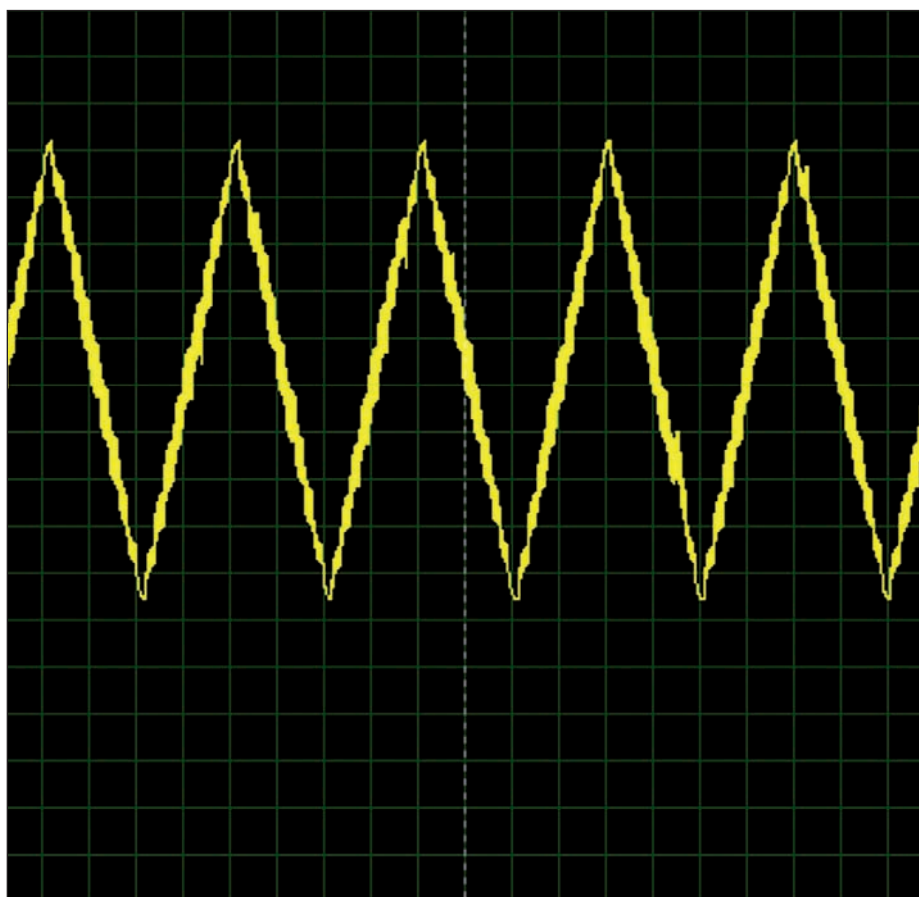



Figura 4: valori di campionamento dell'onda triangolare.

Figura 5: l'onda triangolare a 61 Hz prodotta dal PWM (con filtro RC).

LISTATO 3

```
rem **************************
rem *       LISTATO 3        *
rem *     PWM Hardware       *
rem *  Simulazione Presepe    *
rem *  by Giovanni Di Maria  *
rem **************************
program pwm
dim k as word
main:
     trisc.1=0          'Porta in Output
     portc.1=0          'Azzera porta
     PWM2_Init(1000)    'Inizializza CCP21
     PWM2_Start()
     for k=255 to 0 step -1
         PWM2_Set_Duty(k)   'Cambia duty
         delay_ms(100)
     next k
     PWM2_Stop()
end.
```

filtro RC (R1+C3) che ha il compito di eliminare la parte impulsiva del segnale PWM e trasformare quest'ultimo in componente continua.
Tale filtro, di tipo passa-basso, deve "tagliare" una frequenza che sia almeno la metà di quella dell'onda PWM. In questa maniera viene estratta solamente la componente continua del segnale.

Il listato

Per generare un'onda triangolare con un microcontrollore occorre generare tante tensioni ascendenti e discendenti, di eguali proporzioni. Maggiore è il numero dei campioni generati e più risoluto sarà il segnale prodotto.
Il nostro esempio prevede solo 20 campioni. Ogni campione ha la durata di 500 microsecondi. Un periodo completo, considerando anche il tempo impiegato dalle varie istruzioni, dura 16,39 millisecondi, corrispondenti a 61 Hz.
Se si vuole cambiare frequenza, basta ritoccare i valori all'interno delle due procedure "delay_us".
La generazione degli scalini è resa possibile grazie al cambio continuo del duty cycle del segnale PWM prodotto.
Si può anche ottenere un segnale triangolare più pulito ritoccando leggermente i valori del filtro RC oppure, ancora meglio, aumentando la frequenza del PWM, attraverso la procedura "PWM2_Init(10000)".

Ricordiamo che se aumentiamo anche il numero di campioni, anziché di 20, l'intero segnale sarà qualitativamente migliore.

LUCI CHE DIMINUISCONO PROGRESSIVAMENTE

Come ultimo esempio realizziamo un suggestivo esperimento che può essere utilizzato, ad esempio, nell'allestimento di un presepe natalizio.
Si tratta di una o più luci che abbassano progressivamente la propria luminosità, in un tempo programmabile a piacere. Risulta molto utile nel caso si debba simulare il tramonto del sole o lo spegnersi di una candela.
Lo schema elettrico, pur essendo lo stesso dei precedenti, presenta una novità. Prevede infatti un elemento di pilotaggio di potenza, il mosfet IRF540.
Esso, comandato in tensione con un treno di impulsi di 1 Khz, con un duty cycle sempre più basso, consente di illuminare la lampada con un'intensità via via decrescente, in progressione negativa. Il programma termina quando la lampada si spegne del tutto, segno inconfondibile del tramonto solare avvenuto.

Le impostazioni del firmware determinano una durata dell'intero ciclo di circa 25 secondi. Basterà agire sul livello dello stato di attesa (delay_ms(100)) per aumentare o diminuire tale intervallo.

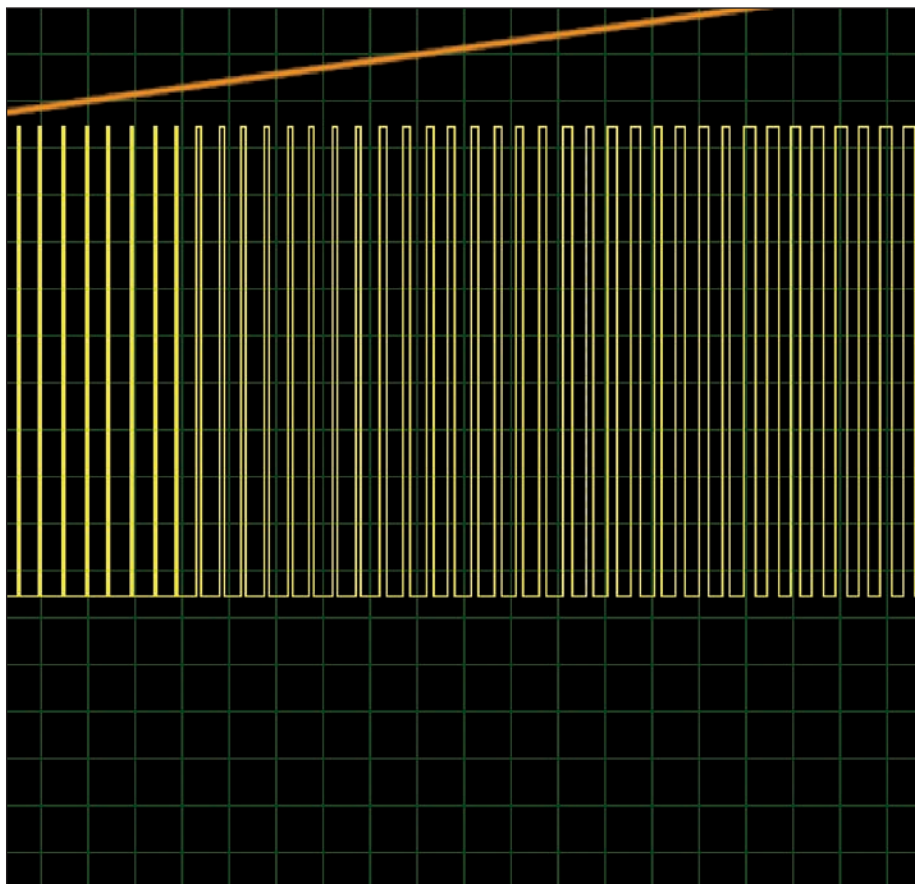



Figura 6: l'onda triangolare a 61 Hz prodotta dal PWM (senza filtro RC).

Figura 7: schema elettrico del simulatore di presepe.

CONCLUSIONI

Dopo aver esaminato a grandi linee le tecniche e le problematiche derivanti dalla generazione dei segnali PWM, possiamo tirare qualche somma.
La tecnica software è più difficoltosa e meno precisa, ma presenta due grandi vantaggi:

- mette a disposizione praticamente tutte le porte del PIC all'uso del PWM;
- può essere implementata in qualsiasi modello di MCU, anche sprovvisto di moduli CCP.

Per contro ha alcuni aspetti collaterali, quali l'occupazione completa del task e una interdipendenza degli eventi esterni dalla stessa generazione. Utilizzare il PWM hardware è invece molto proficuo, anche se la maggior parte dei microcontrollori mette a disposizione solo due moduli. Con essa la memoria del PIC è occupata al minimo, poiché è lo stesso hardware a occuparsi della generazione del treno di impulsi. In più, una volta impartito il comando di inizio PWM, il PIC resta libero e altre istruzioni di programma possono essere inoltrate senza problemi.
Ricordiamo che il segnale PWM è utile in tutte quelle occasioni in cui occorre dosare potenza al carico, senza perdita o dissipazione inutile di calore.
Così, ad esempio, il pilotaggio di lampade di potenza, motori e altri carichi "pesanti", è senz'altro facilitato da tale tecnica. Inoltre, specialmente per il funzionamento dei motori DC, il PWM permette un'ottima coppia motrice, anche a bassi regimi e a basse velocità, scopo non raggiungibile attraverso una limitazione di corrente con limitazione resistiva. ❏

CODICE MIP 2831457

Figura 8: simulatore di presepe.

di ROBERTO NAVONI

VR BRAIN e VR NEURON

LA RISPOSTA ITALIANA AD ARDUINO DUE E RASPBERRY PI

Sviluppate da un'azienda italiana, queste due schede sono largamente impiegate nel settore della robotica, in particolare nel campo degli UAV, ma le loro applicazioni sono davvero molteplici.

Tutto è nato diversi anni fa, quando decisi di comprare il mio primo quadricottero in kit. Sinceramente non credevo fosse in grado di volare, ma in breve tempo ne completai il montaggio ed effettivamente il quadricottero si alzò in volo.

Giocare con questo gioiello di robotica fu sin da subito un'esperienza fantastica, ma il mio spirito tecnico non tardò a manifestarsi facendo nascere in me la curiosità di conoscere tutto sugli algoritmi di volo. Il mio quadricottero era però un sistema chiuso, così iniziai la ricerca in rete di qualche progetto open source e divenni presto parte del team di sviluppo del progetto AeroQuad.

In poco tempo avevo acquisito un'esperienza tale da cimentarmi nella progettazione di una mia scheda, che battezzai MultiPilot 1.0. A supporto della nuova scheda fondai la community VirtualRobotix.com, un luogo in cui utenti e sviluppatori possono scambiarsi esperienze, informazioni e notizie per sviluppare nuove piattaforme di volo, tra cui la MultiPilot32, basata su un microcontrollore STM32.

Dopo tre anni di lavoro sulla piattaforna a 32 bit siamo riusciti a sviluppare la nuova VRbrain, con le seguenti caratteristiche hardware:

- microcontrollore ARM CortexM4F a 168 Mhz con DSP e accelerazione hardware floating-point.

1024 kB di memoria FLASH, 192 kB di RAM;

- accelerometro MEMS, giroscopio, magnetometro e sensore di pressione;
- 6 RC Input standard PPM, PPMSUM, SBUS;
- 8 RC Output a 490 Hz;
- 1 high speed data flash per acquisizione dati;
- 1 Can bus 2 I²C Bus;
- 3 porte seriali disponibili per GPS, telemetria e comunicazione dati;
- 3 digital switch (ULN2003);
- supporto Jtag per l debug realtime della scheda;
- 1 uscita buzzer;
- 1 Input per controllo tensione batterie;
- fornita con modulo GPS compatibile MTEK 3329;
- dimensioni: 60 x 40 x 15 mm.

Figura 1: un quadricottero comandato dal software Arducopter.

Per la VR Brain è disponibile un completo IDE open source che consente la scrittura di codice in C / C++ , la compilazione e il debug inline attraverso il bus JTAG disponibile sulla scheda. All'interno dell'IDE viene impiegato Eclipse come editor, GCC come compilatore e OpenOCD come sistema per il debug realtime. Inoltre è presente un software che consente l'upload direttamente sul micro attraverso la porta USB:

IDE è denominato VRIDE PRO ed è disponbile sul repository opensource del progetto http://code.google.com/p/multipilot32/downloads/list.

Sul repository è disponibile tutto quanto necessario per poter costruire un multicottero con funzionalità avanzate. Il software supportato da VR Brain è l'ultima versione del software Arducopter32 rev 2.8.1.

Il software è implementato con un'architettura modulare che gestisce attraverso un vasta libreria a basso livello diverse tipologie di sensori: accelerometri, giroscopi, magnetometri, GPS, barometri,

Figura 2: vista frontale della scheda VR Brain.

> *La scheda **VRbrain** è disponibile su www.elettrohsop.com/vrbrain-gps.*

radio, uscite per il controllo motore. Tutte queste librerie sono disponibili per realizzare nuovi affascinanti progetti dove l'unico limite è la fantasia.
Arducopter è un progetto che originariamente nasceva per piattaforme Arduino, ma da oltre un anno il nostro gruppo si è impegnato nel porting su piattaforma a 32 bit di tutto il codice sviluppato dalla nostra community e dalla community DIYDRONES fondata da Chris A. negli Stati Uniti. Negli ultimi mesi, vista la continua necessità di risorse avanzate per i nostri droni, anche la community DIYDRONES ha deciso di dare pieno supporto alla nostra piattaforma siglando un accordo di partnership per la produzione e promozione di VR Brain negli Stati Uniti. Per finanziare lo sviluppo della nostra tecnologia, oltre a questo accordo di portata internazionale, abbiamo attivato sul portale indiegogo un progetto di crowdfinance del progetto in modo che chiunque sia interessato a unirsi al nostro progetto potrà farlo acquistando la VR Brain, da semplice utente o da sviluppatore, oppure finanziando il lavoro dei nostri sviluppatori firmware. L'indirizzo del progetto è: http://www.indiegogo.com/vrbrain.
Il firmware Arducopter32 rev 2.8.1 attualmente supporta le seguenti caratteristiche:

- volo acrobatico;
- volo stabile;
- volo semplificato;
- ritorno automatico al punto di partenza;
- mantenimento della posizione e dell'altezza attraverso gps e barometro;
- navigazione automatica su waypoint;
- telemetria attraverso il protocollo Mavlink 1.0;
- gestione di differenti configurazioni Quad, Hexa, Okto;
- controllo automatico della Gimbal.

La VR Brain ha riscosso immediatamente un grande successo e, oltre a far volare multicotteri standard, è stata impiegata per realizzare diversi prototipi, tra cui in paticolare il multicottero ibrido Fox Hybrid, metà aereo e metà multicottero, con incredibili doti di maneggevolezza e controllo sia in volo traslato, in modalità aereo, sia in fase di hovering, in modalità multicottero.
Un altro importate prototipo di multicottero unico al mondo è stato il prototipo di un multicottero a passo variabile con motore unico centrale.
Questo è il link a un video in cui è possibile vedere alcuni test di volo dei nostri prototipi e delle realizzazioni dei membri della nostra community: http://www.virtualrobotix.com/video/uavforge-mov.
Oltre a robot volanti possono essere realizzati anche robot terrestri e marini, come rover, barche a motore o ROV (mini sommergibili). In altri ambiti, come piattaforma di fast prototyping sono stati inoltre realizzati diversi prototipi molto interessanti. Un primo progetto è stato implementato in ambito lighting, utilizzando sensori di illuminazione, suono, PIR e attuatori PWM per pilotare strip a LED o lampade alogene. È stato implementato un sistema automatico che, tramite smartphone interconnesso in bluetooth, consente la gestione completa di una lampada a LED, sia in modalità manuale sia completamente automatica. Un altro progetto molto interessante è stato l'uso di V Brain da parte di Varco Srl per far volare kite e produrre energia elettrica attraverso i venti d'alta quota. VR Brain è stato usato come datalogger a bordo della vela.
Come integrazione alle funzioni di VR Brain negli ultimi mesi abbiamo implementato una nuova piattaforma che abilita l'interconnessione dei nostri robot alla rete tramite la piattaforma VR Neuron.

Figura 3: vista posteriore della scheda VR Brain.

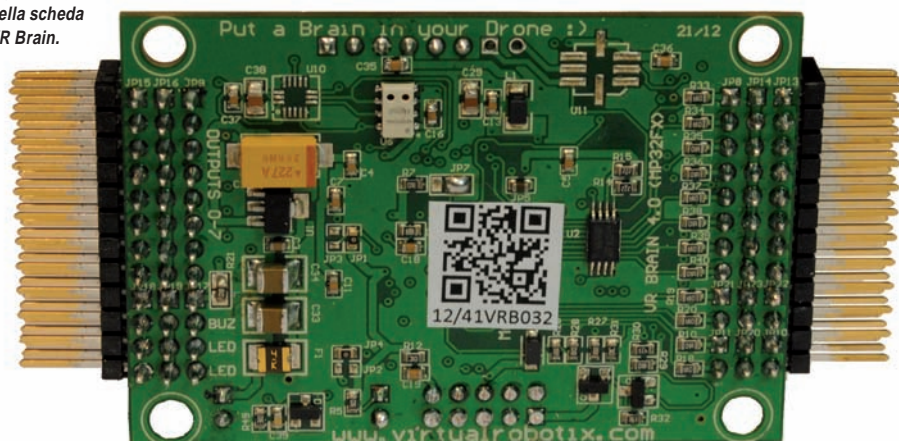
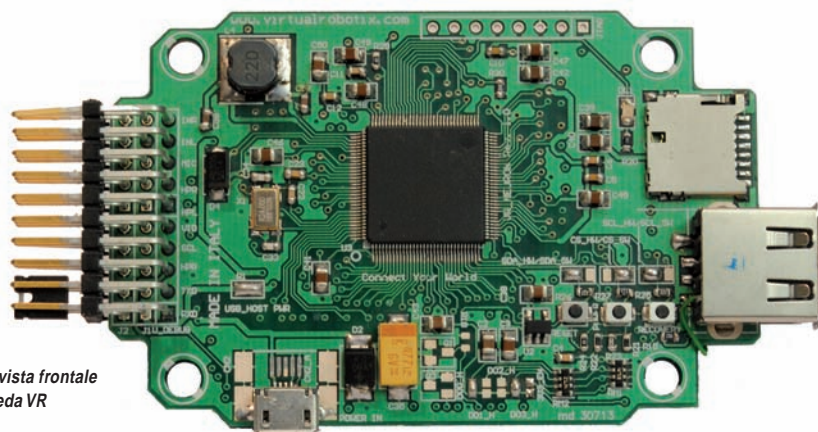



Figura 4: vista frontale della scheda VR Neuron.

VR NEURON

Negli ultimi anni si è assistito all'affermazione di sistemi elettronici di tipo didattico. Le categorie sono fondamentalmente due :
Microcontrollori: rappresentati da schede basate su micro a 8 bit, come Arduino, o a 32 bit, come la VR Brain.
Microprocessori: rappresentati da schede come Beagleboard, Gumstick overo, Olinuxino e, recentemente, RaspberryPI. VR Neuron si pone a meta strada tra le due tecnologie, garantendo la flessibilità e interfacciabilità di un sistema a microcontrollore e la potenza ed espandibilità di un sistema a microprocessore dotato di un sistema operativo potente e scalabile come Linux. Riteniamo che questa nuova scheda possa diventare la componente in grado di consentire ai futuri nuovi designer di interconnettersi a Internet sfruttando al massimo tutte le potenzialità messe a disposizione di processori e sistemi operativi evoluti. Facendo un'analogia con i sistemi organici: il neurone ha un nucleo (la CPU di Neuron), ha delle sinapsi (i device USB di interconnessione: Wi-Fi, Ethernet o 3G) e ha dei dendriti, ovvero le diverse ramificazioni a cui sono collegati altRi neuroni o terminazioni sensoriali che nella Neuron sono rappresentati dai GPIO e i bus di comunicazione. La potenza di Neuron sta nell'essere parte integrante della rete, impiegando le tecnologie di interconnessione più evolute. Da qui la scelta del nome Neuron. Come una famosa frase tratta dalla serie televisiva Star Trek: "Noi siamo Borg, non opponete resistenza; voi sarete assimilati, la resistenza è inutile, Neuron assimilerà le vostre applicazioni, la resistenza e inutile.
Dai microcontrollori, VR Neuron eredita la possibilità di impiegare 26 GPIO , oltre a porte avanzate PWM , porte seriali rs232 , porte di comunicazione ad alta velocità I^{2}C e SPI. Tutto questo consente di impiegare Neuron come se fosse una normale piattaforma Arduino. È possibile interfacciare LED, sensori I^{2}C e SPI e device di tipo seriale ttl.
La scheda può essere allestita con un header a passo 2,5 che si può inserire direttamente su una scheda millefori sulla quale costruire il proprio circuito elettronico. VR Neuron ha però risorse da micro PC, fino a 32 gbyte di storage e 64 mega di memoria DRAM ad alta velocità, con una CPU da 450 Mhz. Numeri di tutto rispetto per una scheda di soli 6x5 cm, la metà di una scheda RaspberryPI, e una porta USB 2.0 standard che consente di interfacciarsi potenzialmente a qualsiasi tipo di device USB standard presente sul mercato. VR Neuron è un dispositivo multimediale entry level e come tale è dotato di un'uscita PAL per interfacciarsi direttamente con display e TV, un'uscita audio stereo e anche di un ingresso microfonico. VR Neuron si può interfacciare a tastiere, mouse, dongle Wi-Fi, Ethernet e webcam, e a qualsiasi altro dispositivo che supporti lo standard USB 2.0.
Per far comprendere le potenzialità di VR Neuron bisogna introdurre una delle componenti fondamentali: il sistema operativo! Dopo diversi test, valutazioni e studi del codice sorgente e dei kernel, abbiamo deciso di impiegare Linux nella sua distribuzione ArchLinuxArm, una distribuzione molto diffusa, ricca di pacchetti preconfezionati e in grado di supportare diverse famiglie di microprocessori arm come v5, v6 , v7. Questa scelta garantirà in futuro la possibilità di portare l'applicazione sviluppata su Neuron anche su piattaforme più evolute.
Segue un esempio applicativo, semplice, ma che mostra la potenza e la flessibilità di VR Neuron.

UNA WEBCAM WI-FI

Attraverso la porta USB 2.0 di VR Neuron è possibile collegare un hub USB a 4 porte sul quale installare un dongle Wi-Fi e una webcam. Il sistema operativo risiede interamente su SD card. Il kernel linux impiegato è la versione 2.6.x su cui sono stati sviluppati specifici driver per supportare dongle USB normalmente disponibili sul mercato, oltre a driver con cui espandere il kernel per supportare, attraverso l'interfaccia standard V4L, le telecamere che supportano lo standard UCV. In questa applicazione la VR Neuron ottiene un indirizzo IP dalla rete attraverso una connessione Wi-Fi e un demone DHCP. Una volta ottenuto l'indirizzo IP viene lanciata l'applicazione mpg_streamer la quale mette a disposizione sul browser degli utenti che si connettono uno streaming video in tempo reale.
Neuron usa il sistema operativo Linux e, quando è connesso, diventa uno strumento di sviluppo pressoché illimitato nelle funzioni e nella scalabilità. La parola chiave è PACMAN, non è il famoso videogioco degli anni Ottanta ma l'acronimo di PACketMANager. Attraverso quest'applicazione dalla potenza impressionante abbiamo la possibilità di scegliere e installare su Neuron fino a 2.350 applicazioni, con semplici istruzioni da riga comando immediatamente utilizzabili e integrabili all'interno della nostra applicazione.
Per interagire con la Neuron, prima di attivare la connessione di rete, si utilizza la porta seriale ttl di debug oppure l'uscita PAL della scheda. Nel mio caso ho usato la seriale ttl per attivare la rete Wi-Fi. Le funzioni impiegate sono quelle standard di Linux per la configurazione di rete, in particolare i comandi ifconfig eiwconfig e i file di configurazione di rete contenuti nella directory /etc/network.d/. Nel file my_wifi_network imposto il SSID della mia rete

domestica, ed eventualmente la password, e configuro la rete in modo tale da impiegare un server DHCP per ottenere un indirizzo IP automatico. Una volta attivata la rete posso iniziare a sfruttare la potenza di PACMAN e inizio a cercare i pacchetti che mi interessano. Attraverso il comando pacman -Ss parola_chiave ho la possibilità di ricercare i pacchetti che più mi interessano, in particolare cerco "webcam". Al comando viene risposta la lista delle applicazioni che soddisfano il criterio di ricerca. Con l'istruzione pacman -Ss mjpg_streamer scarico e installo l'applicazione webcam che mi serve. Tutto questo senza scrivere una riga di codice, ma semplicemente attraverso l'impiego di PACMAN e di un editor come Vi o Nano per editare i file di configurazione delle applicazioni.

Con la stessa logica possono essere implementate applicazioni di terminal server per virtualizzare la porta seriale collegata a Neuron e alla stampante 3D, per esempio!

Penso che uno dei motivi di successo di certe piattaforme di sviluppo siano le cosiddette killer application. Con Neuron, ad esempio, si può controllare a distanza una stampante 3D attraverso la webcam e il terminal server e trasmettere allo smartphone in tempo reale le immagini della lavorazione di una creazione 3D. E magari condividerla con gli amici.

Questo è solo uno dei tanti esempi possibili. Neuron monta un server SSH che consente in qualsiasi momento di collegarsi attraverso un terminale telnet o putty anche con più di una sessione contemporaneamente. Se si vuole usarla come webserver è possibile farlo, oltre che installare Apace2 o Lightppd, usare Python , php e Mysql come database server. Si può utilizzare attraverso un cms per farlo diventare il repository documentale di un'applicazione.

Attraverso il terminale si può interagire con il sistema operativo e lanciare nuove applicazioni, consultare log di sistema o verificare attraverso l'applicazione top lo stato generale del sistema e l'occupazione della CPU, della memoria ecc. Se amate Linux, VR Neuron è la vostra piattaforma di sviluppo hardware!

Se non conoscete ancora Linux dovete immediatamente installarlo sul vostro PC e iniziare a giocarci. Oppure compratevi una Neuron e imparate Linux giocando con questo favoloso nuovo gadget elettronico.

*Vuoi vedere **VRbrain** e **VRneuron** in funzione? Vieni a trovarci il 29 e 30 Gennaio 2013 all'evento "Modena Maker", nell'ambito di Expo Elettronica. Avrai modo di parlare con i progettisti delle schede e assistere a interessanti presentazioni tecniche.*

TOOL DI SVILUPPO

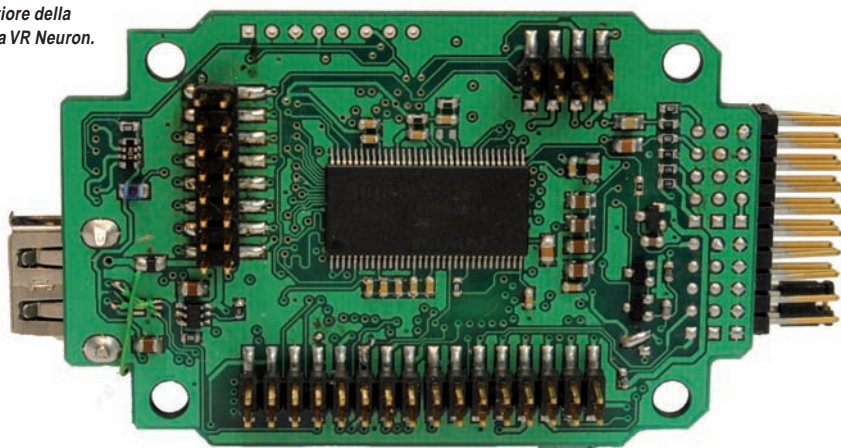
VR Neuron è il prodotto ideale per realizzare molto velocemente prototipi. Attraverso PACMAN è possibile installare il pacchetto base-dev che contiene tutto quanto serve per sviluppare le applicazioni C/C++.

Su Arduino si deve usare un IDE su PC e fare il bootloading dell'applicazione per eseguirla; su VR Neuron si può creare un file con Nano, salvarlo su disco, lanciare la compilazione con il comando gcc e poi eseguire un applicativo asap come task parallelo.

Non c'è dubbio che il nuovo approccio arm + linux sia totalmente rivoluzionario per utenti che provengono dal mondo arduino.

STATO DEL PROGETTO

La scheda è stata sviluppata negli ultimi sei mesi in una ventina di esemplari ed è attualmente in corso la revisione del progetto per passare alla fase di produzione dopo gli ottimi risultati ottenuti nella sperimentazione della scheda. Sono già in corso un paio di design di pre-industrializzazione di prodotti professionali nati sui primi prototipi delle schede.

Figura 5: vista posteriore della scheda VR Neuron.

Entro un mese dovrebbe essere presentato un progetto Kickstarter per raccogliere il budget necessario al primo batch di produzione. In questo momento stiamo organizzando la community per cercare e formare nuovi makers che si uniscano al progetto per implementare e migliorare il kernel di sistema e per implementare template applicativi sulle applicazioni più interessanti per i nostri utenti.

La nostra community è raggiungibile all'indirizzo www.virtualrobotix.com. Periodicamente organizziamo momenti di formazione e di reciproca conoscenza all'interno del polo tecnologico "Kilometro Rosso" a Bergamo, dove ci ospita MPXLAB, un organismo di ricerca di cui facciamo parte e che impiega le nostre schede all'interno di progetti innovativi in ambito professionale ed educativo.

Per il prossimo anno stiamo valutando con partner locali la possibilità di realizzare un fablab dedicato al mondo dei droni: "La Fabbrica dei Droni".

Per maggiori informazioni potete contattarmi direttamente via Skype all'indirizzo virtualrobotix.

PROSSIMAMENTE...

Nei prossimi numeri vedremo come utilizzare la VRbrain e la VRneuron in applicazioni analoghe a quelle in cui viene utilizzato Raspberrt PI o Arduino Due. ❏

CODICE MIP 2831570

di ROBERTO D'AMICO (IW0GTF)

Tutorial PIC18 *(parte settima)*

Un MASTER e uno SLAVE

Vediamo in questa puntata un altro tipo di comunicazione seriale che permette la comunicazione con più dispositivi sulla stessa linea.

Sel precedente numero di Fare Elettronica abbiamo visto come funziona una comunicazione asincrona tra due dispositivi. In questa lezione vedremo invece un altro tipo di comunicazione seriale basata sul BUS I^{2}C, acronimo di Inter Integrated Circuit, utilizzato per la comunicazione tra dispositivi diversi sullo stesso circuito, praticamente un bus di tipo Master/Slave.

I^{2}C

Il bus è stato ideato da Philips nel 1982, per permettere lo scambio di informazioni tra circuiti integrati. Questo BUS di tipo sincrono permette di collegare tra loro più dispositivi. Per funzionare richiede che almeno uno dei dispositivi svolga la funzione di Master, designato alla generazione del clock che serve per sincronizzare la comunicazione tra tutti i dispositivi. Gli altri dispositivi collegati sul BUS sono detti Slave e possono essere più di uno: il limite dipende dalle caratteristiche fisiche della linea che compone il BUS, oltre che dal numero di indirizzi disponibili.

Diversamente da quanto visto nella lezione precedente, dove avevamo solo due attori che dialogavano tra loro (PIC e PC), con il BUS I^{2}C si possono collegare più dispositivi sulla linea. Per poter quindi discriminare tra tutti i dispositivi è quindi necessario associare a ognuno di essi un indirizzo. Considerando che per l'indirizzo sono usati solamente 7 bit, abbiamo 128 indirizzi disponibili, che corrispondono a 128 dispositivi collegabili sulla linea. Di questi però 16 sono riservati, quindi quelli effettivamente disponibili diventano 112.

Le specifiche di questo protocollo, per cui nel tempo sono state definite nuove versioni che hanno apportato miglioramenti, prevedono anche la possibilità di avere sulla stessa linea più dispositivi Master.

Il protocollo, di tipo seriale, utilizza due fili per il collegamento tra i dispositivi. Uno

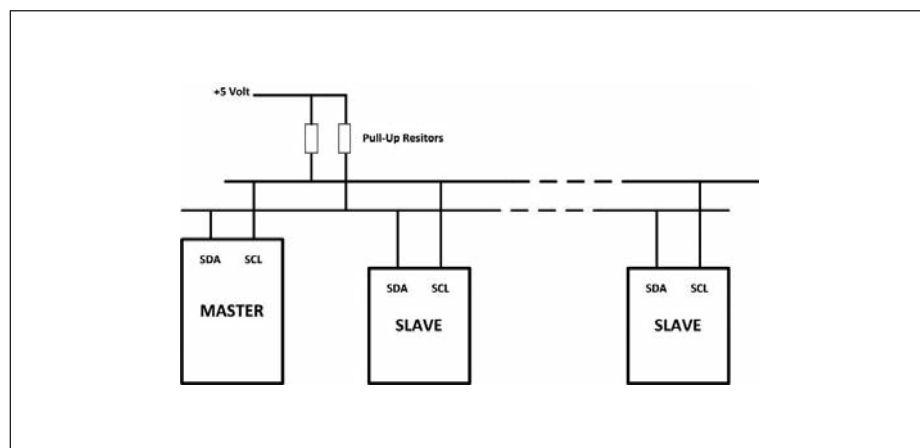



Figura 1: schema di collegamento tra dispositivi.

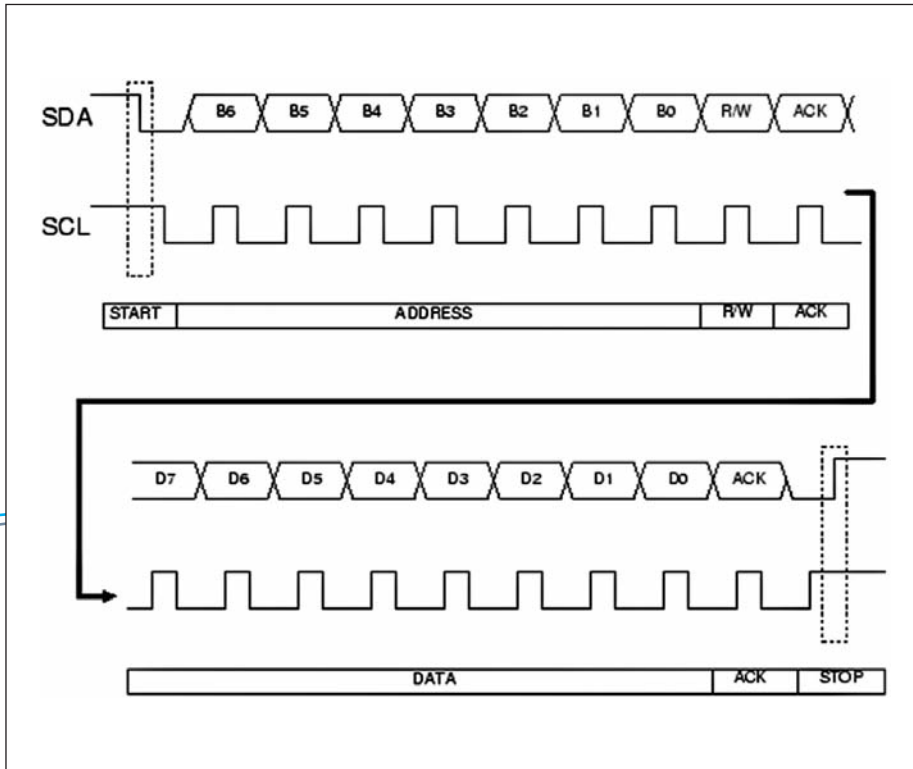



Figura 2: flusso dei dati di un messaggio.

è il clock (SCL - Serial Clock Line), l'altro è utilizzato per lo scambio dei dati (SDA - Serial DAta line). Entrambe le linee sono mantenute sempre a livello alto tramite due resistenze di pull-up, il cui valore è utilizzato anche per variare l'impedenza della linea.

La comunicazione parte sempre da un Master il quale porta la linea dei dati a livello basso, mentre quella del clock rimane alta grazie alla resistenza di pull-up. Questa condizione indica lo START della comunicazione.

La comunicazione da parte del Master continua portando a livello basso anche la linea del clock, quando il segnale del clock torna a livello logico alto, il valore presente sulla linea dati può essere considerato valido e quindi acquisito, continuando a inviare i dati fino alla condizione di STOP in cui la linea di clock è tenuta alta e quella dati passa dallo stato basso ad alto.

Quando il Master avvia una comunicazione, il primo dato inviato è l'indirizzo del dispositivo che sta interpellando e l'ottavo bit serve a indicare se il master vuole scrivere o leggere informazioni dal dispositivo. Nel caso in cui il master richieda informazioni allo slave, il primo rilascerà il controllo della linea dati che a quel punto sarà gestita dallo slave per inviare i dati, sempre sincronizzandosi con il clock che è generato dal master.

SRF02, UN SENSORE A ULTRASUONI

Come dicevamo in precedenza esistono diversi circuiti integrati che fanno uso di questo protocollo, ma si possono trovare anche altre tipologie di dispositivi. Tra questi troviamo anche telemetri a ultrasuoni come lo SRF02. Questo sensore permette di compiere misurazioni di distanze in un intervallo che va da 16 centimetri a 6 metri.

Il sensore utilizza il protocollo I^{2}C e funziona da Slave con un proprio indirizzo. Come abbiamo visto nel paragrafo precedente questo BUS permette il collegamento di più dispositivi sulla stessa linea. Tra le specifiche del sensore sono riportati gli indirizzi che è possibile utilizzare

LISTATO 1

```
#include <p18f2620.h>
#include <delays.h>
#include <i2c.h>
/**********
 * DEFINE *
 **********/
#define LED1 LATAbits.LATA2
#define LED2 LATAbits.LATA3
#define LED3 LATAbits.LATA4
#define LED4 LATAbits.LATA5
#define PULSANTE1 PORTBbits.RB0
#define PULSANTE2 PORTBbits.RB1
#define SRF02 0xE0 /* Indirizzo del SRF02 */
/************
 * FUNZIONI *
 ************/
unsigned int srf02_read(void)
{
    int result=1;
    unsigned int measure=0;
    unsigned char measurement[2];

    result = EEByteWrite(SRF02, 0x00, 0x51);
    /*Verifica l’esito dell’operazione di avvio della lettura*/
    if(result==0)
    {
        /*Attende il completamento della misurazione del sensore*/
        Delay10KTCYx(70);
        /*Recupera la misurazione fatta dal sensore */
        result = EESequentialRead(SRF02, 0x02, measurement, 2);
        /*Verifica l’esito dell’operazione di lettura del sensore*/
        if(result==0)
        {
            /*Scrive il valore letto dal sensore*/
            measure = (measurement[0]<<8);
            measure += measurement[1];
        }
    }

    return measure;
}
/********
 * MAIN *
 ********/
void main(void)
{
    int distanza=0;
    /*Impostazione dei registri di configurazione del PIC*/
    ADCON1=0b00001111;
    CMCON=0b00000111;
    /*Reset iniziale delle porte*/
    PORTA=0;
    PORTB=0;
    PORTC=0;
    /*Definizione della funzione delle singole porte*/
    TRISA=0x03;
    TRISB=0b00000011;
    TRISC=0x00;
    /*Configurazione del modulo MSSP*/
    OpenI2C(MASTER, SLEW_OFF);
```

Addio 7805 e 7905

I modelli switching equivalenti sono migliori

I regolatori lineari di tensione, di tutte le forme e dimensioni, sono utilizzati in un numero enorme di circuiti elettronici. Dispositivi classici come il 7805 con uscita positiva e il 7905 con uscita negativa sono indispensabili nella maggior parte dei circuiti, ma essi non sono molto efficienti dal punto di vista energetico a causa della loro modalità di funzionamento lineare. Tutta la potenza superflua tra i pin di ingresso e di uscita di questi regolatori viene infatti convertita in calore. Sicuramente oggi possiamo fare di meglio: maggiore efficienza energetica e maggiore rispetto per l'ambiente? Questo è proprio l'argomento di questo articolo. Il messaggio è chiaro: abbasso la dissipazione di potenza, evviva lo switching!

LISTATO 1 segue

```
    SSPADD=0x63; //100kHz BUS @ 40MHz PIC
    /*Accende il LED4 per indicare che il programma è avviato*/
    LED1=1;
    LED2=1;
    LED3=1;
    LED4=1;
    Delay10KTCYx(250);
    LED1=0;
    LED2=0;
    LED3=0;
    LED4=0;

    // Loop infinito
    for(;;)
    {
        distanza=srf02_read();
        /* Accende i led in funzione della distanza */
        if(distanza>20 && distanza<=40)
        {
            LED1=1;LED2=0;LED3=0;LED4=0;
        }
        if(distanza>40 && distanza<=60)
        {
            LED1=1;LED2=1;LED3=0;LED4=0;
        }
        if(distanza>60 && distanza<=80)
        {
            LED1=1;LED2=1;LED3=1;LED4=0;
        }
        if(distanza>80)
        {
            LED1=1;LED2=1;LED3=1;LED4=1;
        }
    }
}
```

per questo sensore. Quelli disponibili sono 16, che corrisponde al numero massimo di dispositivi che possono essere utilizzati sulla stessa linea.
Il sensore funziona come una memoria E2PROM, si devono quindi indirizzare le locazioni di memoria interna per gestirlo. Per avviare una misurazione in centimetri si deve scrivere il comando 81 nella locazione di memoria 0. La lettura viene completata in 66 ms, dopo di che si possono leggere i due byte nella locazione 2 e 3 della memoria del sensore, che corrispondono rispettivamente alla parte alta e alla parte bassa del numero che indica la distanza misurata.
Chiudiamo questo paragrafo con una curiosità: questo sensore utilizza un PIC per eseguire le misurazioni con gli ultrasuoni e per gestire le comunicazioni come Slave I^2C.

IL CIRCUITO

Le modifiche allo schema base di questo corso sono molto poche, si deve collegare solamente il sensore SRF02 al PIC collegando tra loro i pin SDA e SCL dei due componenti, che sul PIC corrispondono hai pin RC4 e RC3 rispettivamente, mentre sul sensore sono i pin 2 e 3.
A entrambi questi collegamenti andrà aggiunta una resistenza, che dall'altro capo sarà collegata verso l'alimentazione positiva. Questo collegamento, definito di pull-up, serve per mantenere sempre alto il livello logico dei segnali della linea. Il sensore richiede anche i collegamenti per l'alimentazione, quindi per la massa e i +5 volt, mentre, come potete notare nello schema in **figura 4**, il pin 4 del sensore non va collegato.
Il sensore SRF02 può funzionare in due modalità di comunicazione: in I^2C, che utilizziamo in questa lezione, oppure con un classico collegamento seriale, come quello visto nella lezione precedente. Per cambiare la modalità di funzionamento si usa il pin 4 che, lasciato libero, fa funzionare il circuito in modalità I^2C, mentre collegato va verso massa.

IL MODULO MSSP

Il modulo Master Synchronous Serial Port propone un'ulteriore interfaccia seriale, oltre a quella EUSART vista nel precedente numero di Fare Elettronica, che può operare in due modalità SPI e I²C. Per quest'ultima, in particolare, il modulo hardware presente nel PIC può funzionare nelle modalità Master, Multi-Master e Slave.

In questa lezione, come abbiamo visto, prendiamo in esame solo la modalità Master per controllare il sensore SRF02 che sarà il dispositivo Slave.

Questo modulo utilizza tre registri base di configurazione e il controllo del funzionamento della periferica. Utilizzati in entrambe le modalità di funzionamento del modulo, essi sono SSPCON1, SSPCON2 e SSPSTAT.

Figura 3: foto del sensore SRF02.

Per l'utilizzo del modulo in modalità I²C vengono utilizzati altri tre registri. Uno di questi è SSPSR (MSSP Shift Register), e si tratta di un registro a cui non è possibile accedere ma viene usato dal modulo per scomporre i byte dei dati da inviare in singoli bit o, viceversa, per ricostruire il byte dai singoli bit ricevuti sulla linea dati.

L'altro registro è SSPBUF (MSSP Buffer), sostanzialmente il buffer per i dati che vengono ricevuti o devono essere trasmessi.

Lo scopo dell'ultimo registro, SSPADD (MSSP Address), dipende dalla modalità in cui è configurato il modulo. Quando questo funziona come Slave, in questo registro va memorizzato l'indirizzo a cui deve rispondere, mentre nel caso in cui il

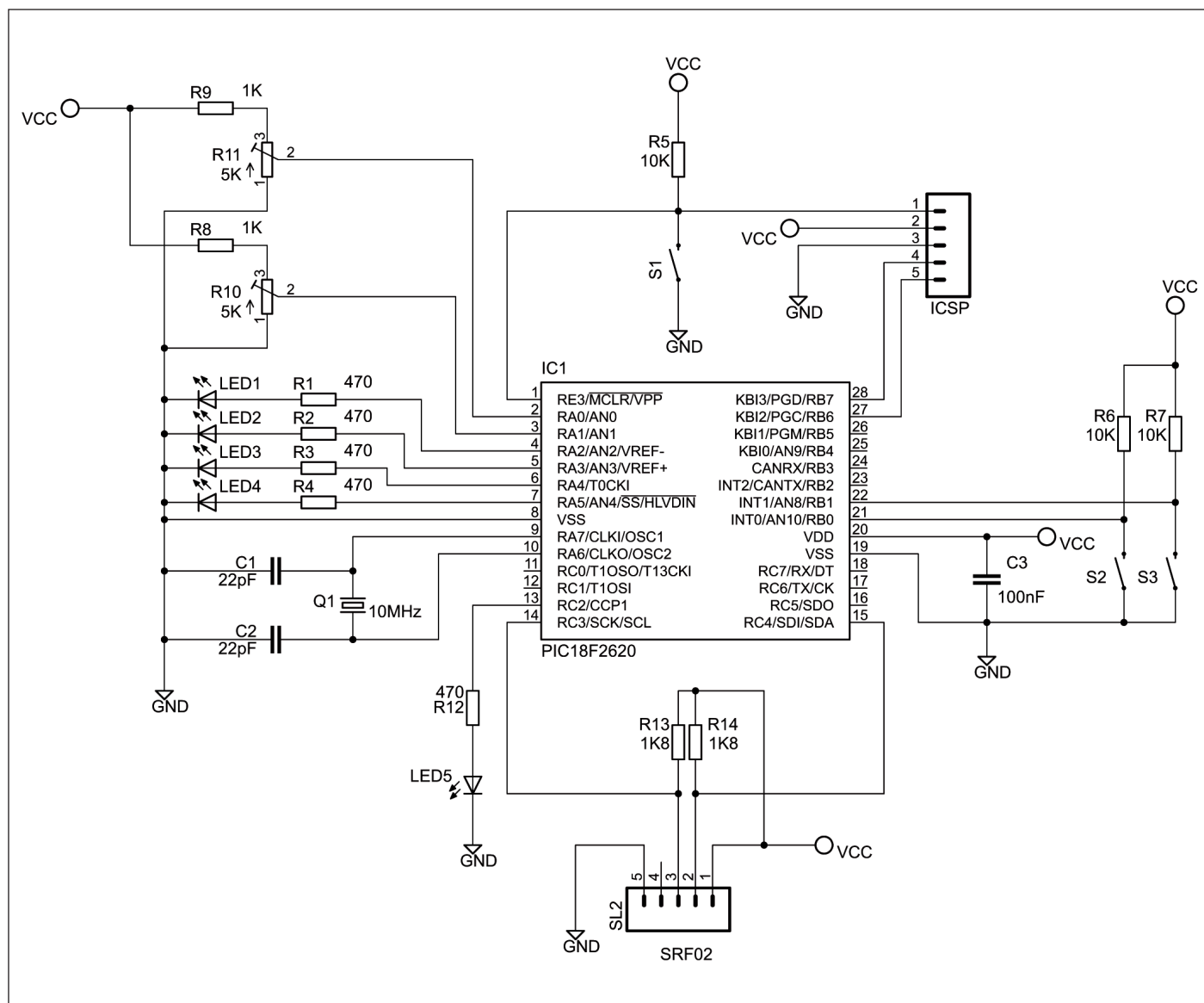



Figura 4: lo schema elettrico del circuito.

modulo venga utilizzato come Master, esso serve per determinare la frequenza del clock di comunicazione sul BUS I^2C. Data la complessità del protocollo questa volta non andremo ad approfondire il funzionamento e la configurazione di tutti i registri, rimandando a chi fosse interessato alla lettura del datasheet. Nel programma di esempio andremo quindi a utilizzare le funzioni del C18 per la gestione di questo modulo. L'unico parametro che dovremo configurare è il registro SSPADD per impostare la frequenza di funzionamento della linea I^2C. Per farlo, ancora nel datasheet troviamo una tabella dove per le varie configurazioni di clock del PIC e della frequenza desiderata per il BUS sono riportati i corretti valori per il registro.
Nel caso del circuito illustrato in questa lezione abbiamo un PIC che funziona con una frequenza di 40 MHz e un sensore (SRF02) che può lavorare a una frequenza di 100 KHz sulla linea I^2C. Quindi, incrociando questi dati sulla tabella, troviamo il valore 0x63 da inserire nel registro SSPADD.

MISURIAMO LE DISTANZE

Il programma che analizzeremo stabilisce la comunicazione con il sensore, al quale ciclicamente richiede di effettuare una misurazione di distanza in funzione della quale procede all'accensione dei quattro LED del circuito.
Osservando il **listato 1** troviamo subito nelle prime righe l'inclusione di un nuovo file, i2c.h, nel quale sono definite funzioni ed etichette per la gestione del modulo MSSP in modalità I^2C. Nelle righe che seguono è stata aggiunta la definizione della label SRF02, a cui è associato l'indirizzo del sensore. Come al solito nel listato di esempio non sono riportati tutti i pragma per la configurazione del PIC, che trovate comunque nel codice scaricabile dal sito di Fare Elettronica.
Come potete notare, all'interno del main, nella configurazione delle porte tramite i registri TRISx, le porte del registro C sono configurate tutte come out, anche se il pin RC4 dedicato alla linea dati funzionerà anche come ingresso quando il PIC dovrà ricevere le informazioni dal sensore, questo perché sarà direttamente il modulo MSSP a occuparsi del cambio di direzione della porta.
Proseguendo troviamo la chiamata alla funzione OpenI2C, la quale richiede due parametri: il primo indica il modo di funzionamento, quindi per questo esempio sarà MASTER, mentre il secondo indica che lo Slew Rate deve essere disabilitato

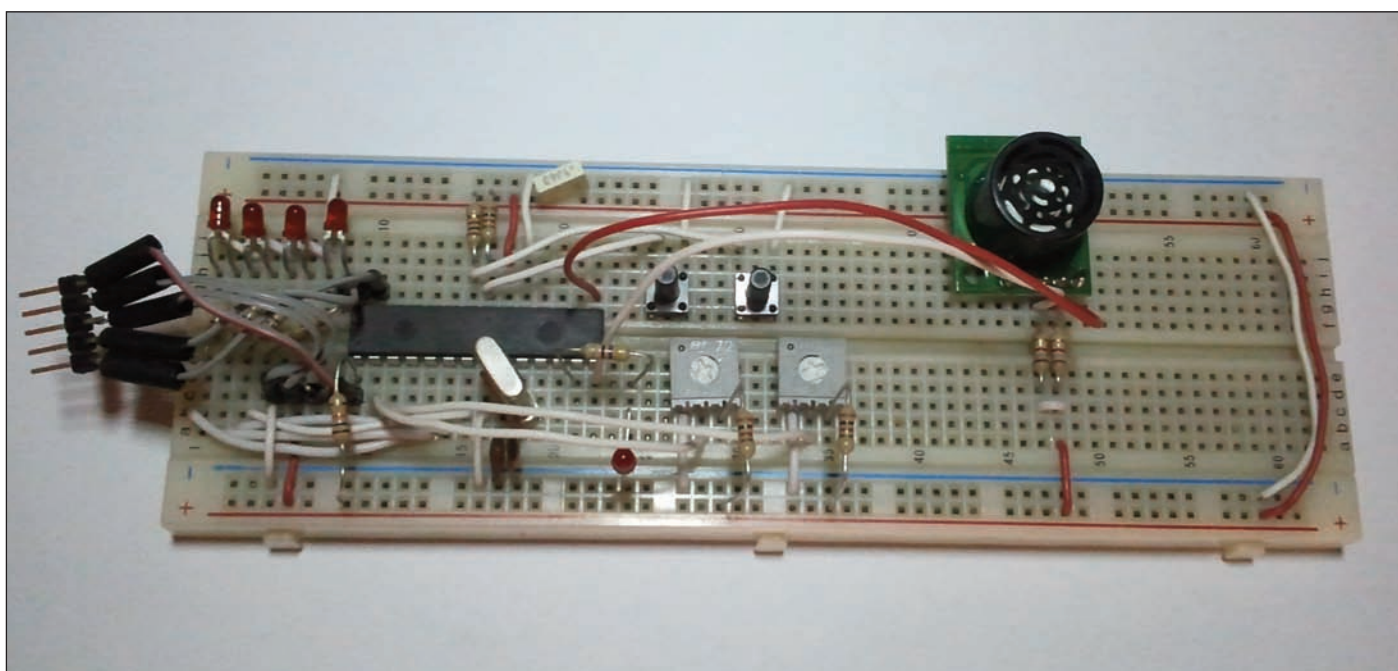



Figura 5: Il circuito montato sulla bread-board.

MO.MA

PER *approfondire...*

- http://it.wikipedia.org/wiki/I%C2%B2C
- http://www.robot-electronics.co.uk/htm/srf02techI2C.htm

perché utilizziamo una frequenza del BUS di 100 kHz.
Per completare la configurazione del modulo MSSP, come abbiamo visto anche in precedenza, si deve impostare il registro SSPADD con il corretto valore per generare la giusta frequenza del clock del BUS. Questo si ottiene semplicemente assegnando al registro il valore 0x63.
La parte principale del programma, all'interno del loop, richiama la funzione srf02_read, di cui parleremo dopo e, in funzione della distanza misurata, accende uno o più LED applicando semplici controlli, eseguiti con una sequenza di if, su questo valore.
Vediamo ora la funzione srf02_read. Come abbiamo già detto essa serve a far eseguire al sensore una misurazione della distanza e ritorna, per com'è stato scritto il programma, un numero che rappresenta i centimetri misurati. Internamente alla funzione sono definite tre variabili: result di tipo intero è utilizzata per memorizzare l'esito della risposta delle funzioni che scrivono e leggono le informazioni sul sensore; measure è un intero senza segno utilizzato per contenere il valore finale della misura; measurement rappresenta un array di due campi di char senza segno, utilizzato come buffer per memorizzare i due byte letti sul sensore che rappresentano la misurazione eseguita. Come abbiamo detto in precedenza, il sensore si comporta come una memoria EEPROM I^{2}C.
Andiamo quindi a utilizzare le funzioni di libreria specifiche per il controllo di questo tipo di dispositivi. La prima ad essere richiamata è EEByteWrite. Come abbiamo visto nel paragrafo riguardante il sensore SRF02, le misurazioni sono effettuate a richiesta e per fare questo si deve scrivere alla locazione di memoria interna del sensore 0x00 il comando che si vuole usare.
Nel **Listato 1** è usato il valore 0x51 per ottenere il risultato in centimetri, quindi alla funzione EEByteWrite passeremo come parametri l'indirizzo che identifica il sensore sul BUS I^{2}C, l'indirizzo della locazione di memoria interna e il dato da scrivere in questa locazione. La funzione restituisce 0 se l'operazione di scrittura è andata a buon fine, e questo valore è memorizzato nella variabile result e controllato tramite un if. Nel caso in cui sia restituito un valore diverso da 0, la funzione termina restituendo il valore della variabile measure che, non essendo stato cambiato, sarà 0. In questo modo in caso di errori avremo una misurazione "nulla" con un valore che possiamo anche non considerare. Il passo successivo è attendere 70 ms per essere certi che l'operazione di misurazione sia stata completata e poi andare a recuperare il risultato della stessa tramite la funzione EESequentialRead, che legge dal dispositivo, a partire dalla locazione di memoria interna 0x02, un numero di byte indicati dall'ultimo parametro, in questo caso 2, e li memorizza nella variabile measurement. Anche in questo caso la funzione restituisce un valore 0 se tutto è andato bene, quindi solo in questo caso i due byte ricevuti vengono "uniti" nella variabile measure per restituire la misurazione in centimetri effettuata dal sensore.

CONCLUSIONI

In questa lezione abbiamo avuto modo di vedere come un BUS, basato su una linea seriale sincrona, ci permetta di collegare tra loro più dispositivi in grado di scambiarsi informazioni. Ricordo che esiste una grande varietà di dispositivi che possono essere controllati con il protocollo I^{2}C. Per fare qualche esempio: display, memorie, convertitori AD, I/O expander e altri. Come possibile esercizio di approfondimento potete provare a realizzare un mix di questa lezione con la precedente. Come avrete ormai capito i moduli utilizzati sono diversi e possono quindi essere utilizzati contemporaneamente. È possibile quindi compiere misurazioni con il sensore e inviare i dati al PC tramite comunicazione seriale RS232 per visualizzare in modo più accurato le misurazioni di distanza eseguite. ❑

CODICE MIP 2821447

Protocol Simulator Board

Problemi per analizzare i segnali BUS presenti nel proprio progetto?
Zeroplus viene in aiuto con questa scheda per l'analisi dei protocolli!
Durante lo sviluppo o la decodifica dei segnali, la scheda "PSB" aiuta a visualizzare i pacchetti e ridurre così i tempi di sviluppo migliorando l'efficienza. Lavorando con il Logic Analyzer, si possono valutare le relazioni della base di tempo e dei pacchetti migliorando il progetto.
Ideale sia per i progettisti professionisti che per la didattica.

CODICE MIP 2831799

PC System	Memory	Digital Audio	IC Interface	Optoelectronics	Wireless	Other
Compact Flash 4.1	1-Wire, 3-Wire	AC97	HPI	7-SEGMENT LED	Differential Manchester	BDM, DS 1302
eMMC	I2C	HD Audio	JTAG 2.0	CCIR656	DigRF	DS18B20
FWH	I2C(EEPROM 24L)	HDMI CEC	MCU-51 DECODE	COME IMAGE	ISO7816 UART	DSA Interface
Low Pin Count	I2C(EEPROM 24L	I2S	MICROWIIRE	DALI Interface	KEELOQ Code Hopping	GPIB, HART
LPC-SERIRQ	CS6/24LCS62)	PCM	SIGNIA 6210	DM114/DM115	MANCHESTER	ModBus
LPT, PCI, PECI	MICROWIRE(EEP	S/PDIF	SLE4442	DMX512	MII	MODIFIED SPI
PS/2	ROM 93C)	STBus	SSI Interface	LCD12864	MIL-STD-1553	OPENTHERM 2.2
DS2.0/DSIO, SVID	SPI, SPI PLUS		ST669	LCD1602	MILLER	PSB Interface
UART(RS232C	SPI(EEPROM AT25F)			LG4572	MODIFIED MILLER	PROFIBUS
/422/485)	SAMSUNG			S2Cwire/AS2Cwire	SWP	SHT11, YK-5
USB 1.1	K9(NAND flash)			SCCB	WIEGAND	1-Wire(Advanced)
USB 2.0	UNI/O				WWV/WWVH/WWVB	

Infrared rays	Basic Logic Application	Power	Automotive
IRDS	ARITHMETICAL LOGIC	BMS, HDQ	CAN 2.0B
NEC PD6122	DIGITAL LOGIC	PMBus 1.1	DSI Bus
Philips RC-5	JK FLIP-FLOP	SDQ	FlexRay 2.1A
Philips RC-6	UP DOWN COUNTER	SMBus 2.0	LIN 2.1
PT2262/PT2272			MVB
			WTB

LAP-C Series

LAP-B Series

Per maggiori informazioni www.elettroshop.com/psb oppure chiamare lo 02/66504794

di GIOVANNI DI MARIA

Corso mikroBASIC

La libreria OneWire: MISURIAMO LA TEMPERATURA

Eccoci arrivati all'ultima puntata del corso su mikroBasic. In questo articolo scopriremo come utilizzare la libreria 1-Wire con il PIC, sperimentando alcune applicazioni pratiche con il termometro digitale DS1820.

La libreria OneWire, conosciuta anche con il nome 1-Wire, permette la comunicazione tra dispositivi prodotti da Dallas. È un tipo di comunicazione, instaurata su un unico filo, di tipo bidirezionale, tra un master e uno o più slave. Ogni dispositivo di tipo slave contiene al suo interno, memorizzato su ROM, un codice univoco a 64 bit. Questa univocità permette al master di selezionare un determinato slave (tra molteplici), presenti sulla linea. Se si utilizza un solo dispositivo sulla stessa linea, non c'è bisogno di leggere tale codice seriale.
I dispositivi utilizzano valori di tensione di alimentazione compresi tra 3 e 6 volt. La trasmissione è di tipo bidirezionale. Essa è di tipo half-duplex, ossia la ricezione non avviene contemporaneamente alla trasmissione, e viceversa.

IL DS1820, SENSORE DI TEMPERATURA DIGITALE 1-WIRE

Il termometro DS1820 è un sensore di temperatura di tipo 1-Wire. Esso può rilevare temperature con precisione di 9 bit, in un range compreso tra –55 °C e +125 °C, con un'approssimazione di +/– 0,5 °C. La particolarità di questi sensori di temperatura risiede nel fatto che ciascun dispositivo ha un numero seriale unico, di 64 bit, che lo identifica univocamente. Tale ID è memorizzato all'interno di una ROM presente all'interno. Questa caratteristica permette di poter utilizzare un vasto numero di sensori su un unico bus di dati. Il protocollo utilizzato da questi sensori è 1-Wire; questo tipo di comunicazione richiede un unico pin per il suo funzionamento. Inoltre, per poter funzionare correttamente, questi sensori di temperatura non necessitano di alcun altro componente esterno. Possono essere alimentati direttamente dalla linea dati e il range di alimentazione supportato è compreso tra 3 e 6 V.

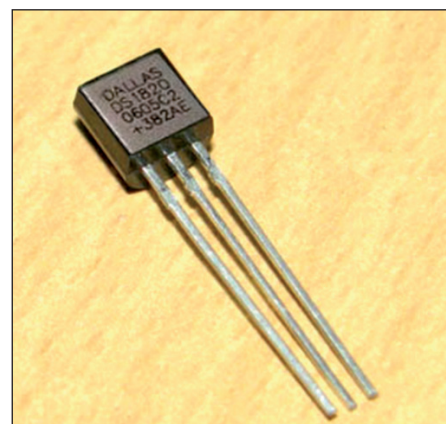



Figura 1: il termometro DS1820.

L'alimentazione elettrica può essere attinta direttamente dalla sua linea dei dati, cosicché esso non ha la necessità di alimentazione esterna.

Caratteristiche del DS1820

Ecco le principali caratteristiche:
- richiede solo un filo per la comunicazione;
- non necessita di componenti esterni;
- misura temperature da –55 °C a +125 °C con incrementi di 0,5 °C. In scala Fahrenheit corrisponde a un intervallo compreso tra –67 °F e +257 °F con incrementi di 0,9 °F;
- la temperatura è letta come valore digitale a 9 bit;
- la conversione della temperatura in word digitale avviene in soli 200 ms.

LA LIBRERIA ONEWIRE DEL MIKROBASIC

Il mikroBasic dispone di una versatile libreria per gestire i dispositivi 1-Wire. Uno di questi è, appunto, il termometro

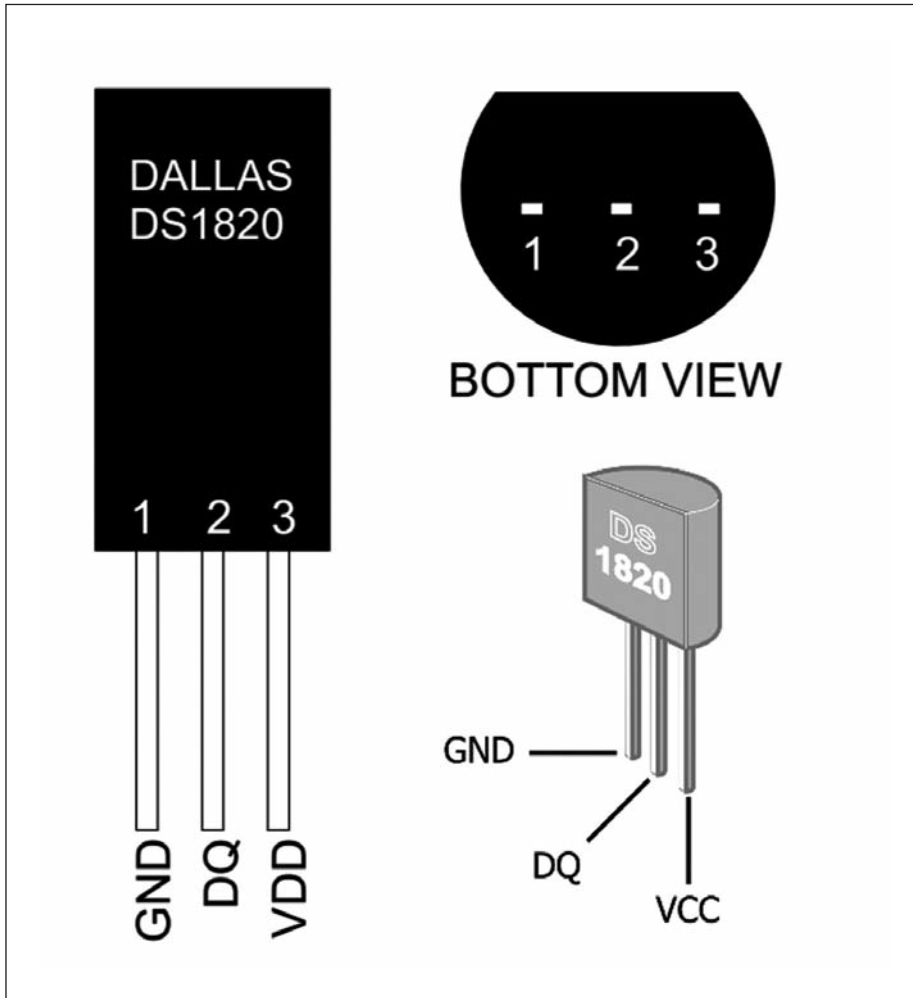



Figura 2: pinout del DS1820.

DS1820. Questo tipo di protocollo, come detto, prevede la presenza di un Master e uno o più Slave, utilizzando un solo filo. I dispositivi 1-Wire sono a collettore aperto (open collector) e devono utilizzare una resistenza di pull-up sulla linea.
Ogni dispositivo dispone di un numero di registrazione unico che lo identifica dagli altri. In questo modo è possibile collegarne più di uno, sulla stessa linea.

L'identificazione è formata da:
- 8 bit che identificano il tipo di dispositivo;
- 48 bit che identificano il numero seriale;
- 8 bit per il CRC.

Per l'utilizzo del termometro Dallas DS1820 è importante tenere a mente due importanti fattori:
- l'oscillatore deve avere una frequenza minima di 4 Mhz, al fine di utilizzare correttamente le funzioni per la gestione dei dispositivi;
- occorre disabilitare gli interrupt per utilizzare la libreria 1-Wire. Questo perché essa implementa una programmazione a basso livello delle temporizzazioni interne del microcontrollore.

LE FUNZIONI E LE PROCEDURE IN MIKROBASIC

In dettaglio adesso esploriamo le funzioni e le procedure che gestiscono totalmente il protocollo 1-Wire. Esse sono di numero molto limitato e il loro utilizzo è estremamente semplice. Le funzioni o procedure sono le seguenti:
- Ow_Reset;
- Ow_Read;
- Ow_Write.

Ow_Reset

È una funzione che invia un segnale di reset al dispositivo 1-Wire. Essendo una funzione, ritorna un valore numerico di tipo byte che può rappresentare due possibili situazioni:
- 0, se il dispositivo viene trovato ed è, quindi, presente;
- 1, se il dispositivo non è presente.

Sintassi:
```
sub function Ow_Reset(dim byref
port as byte, pin as byte) as
byte
```

dove:
- "port" rappresenta la porta dove è collegato il dispositivo;
- "pin" rappresenta il pin dove è collegato il dispositivo.

Esempio di utilizzo

L'esempio che segue resetta il termometro collegato sulla linea RE2:

```
Ow_Reset(PORTE, 2)
```

Ow_Read

È una funzione che legge un dato dal dispositivo collegato sul bus.

Sintassi:
```
sub function Ow_Read(dim byref
port as byte, dim pin as byte) as
byte
```

dove:
- "port" rappresenta la porta dove è collegato il dispositivo;
- "pin" rappresenta il pin dove è collegato il dispositivo.

Esempio di utilizzo

L'esempio che segue legge un byte dal dispositivo:

```
tmp = Ow_Read(PORTE, 2)
```

Ow_Write

È una procedura che scrive un dato sul bus 1-Wire.

Figura 3: schema elettrico di principio.

Figura 4: lo zoccoletto per ospitare il DS1820 sulla EasyPIC.

Sintassi:

```
sub procedure Ow_Write(dim byref
port as byte, dim pin, par as
byte)
```

dove:

- "port" rappresenta la porta dove è collegato il dispositivo;
- "pin" rappresenta il pin dove è collegato il dispositivo;
- "par" rappresenta il dato da scrivere sul bus.

Esempio di utilizzo

L'esempio che segue scrive un byte sul dispositivo:

```
Ow_Write(PORTE, 2, $CC)
```

SCHEMA ELETTRICO DI PRINCIPIO

Il collegamento del termometro al microcontrollore è estremamente semplice. Utilizzando buone connessioni si può raggiungere una distanza ragguardevole di parecchi metri.

Il DS1820 è alimentato al pin 3 ed è collegato a massa attraverso il pin 1. La linea dati è al centro dei contatti, corrispondente al pin 2. Tale linea si deve trovare collegata a VCC, attraverso una resistenza di pull-up da 4,7 kohm oppure da 10 kohm, essendo di tipo "open collector". Non sono necessari altri collegamenti o altri dispositivi, se non quelli per la visualizzazione dei dati come, ad esempio, un display LCD o dei display a 7 segmenti.

IL DS1820 SULLA EASYPIC

I possessori della scheda di sviluppo EasyPIC saranno contenti di sapere che essa è predisposta a ospitare il termometro DS1820. Già dalla obsoleta versione 2 della scheda è presente uno zoccoletto a tre piedini per la collocazione dell'integrato. Tale zoccoletto, contrassegnato con TS1, è dotato di tre fori per permettere l'alloggiamento del termometro. Sulla scheda di sviluppo è indicata altresì la forma dell'integrato, al fine di orientarlo e posizionarlo correttamente. La sua comunicazione verso il microcontrollore è gestita dal jumper J11, che ne coordina anche la disconnessione oppure il collegamento alle porte RE2 o RA4.

COME RILEVARE LA TEMPERATURA CON IL DS1820

Il termometro DS1820 non fornisce una tensione analogica, come il suo fratello minore LM35. Il suo funzionamento è squisitamente digitale e la comunicazione avviene mediante un "botta e risposta" di comandi inviati e ricevuti. Anche la codifica delle informazioni (temperatura) è totalmente digitale e il calcolo deve avvenire tenendo contro proprio di questo aspetto. A grandi linee, ecco quale è la procedura da seguire, per poter rilevare una temperatura. Si suppone che il DS1820 sia correttamente collegato e funzionante:

- reset del termometro. Corrisponde ad un segnale logico di livello basso della durata di almeno 480uS;
- scrittura del comando esadecimale 0xCC. Serve per bypassare la lettura della ROM, se si utilizza solamente un dispositivo;
- scrittura del comando esadecimale 0x44. Serve per iniziare la procedura di conversione;
- reset ulteriore del termometro;
- scrittura del comando esadecimale 0xCC;
- scrittura del comando esadecimale 0xBE. Serve per predisporre alla ricezione della temperatura a 9 bit;
- prima lettura del dispositivo, al fine di acquisire la temperatura vera e propria (LSB). Il dato, rappresentato in 8 bit, è differente a seconda se la temperatura sia

convertitori USB-seriale

compatti, performanti e convenienti

CODICE MIP 2831803

Per maggiori informazioni visita Il sito **www.elettroshop.com** oppure chiama il numero **02/66504794**

positiva o negativa (sopra lo zero o sotto zero);

- seconda lettura del dispositivo, al fine di acquisire il segno della temperatura (MSB). Sebbene tale informazione sia solamente un bit, esso è ripetuto a tutto il byte letto. Nella fattispecie, una valore binario di %00000000 rappresenta una temperatura positiva o di zero, mentre un valore binario di %11111111 ($FF) rappresenta una temperatura negativa.

Ricordiamo che il termometro DS1820 misura le temperature in un intervallo compreso tra –55 °C e +125 °C, con incrementi di 0,5 °C (mezzo grado).

Come si nota dalla **tabella 1**, il valore digitale restituito dal termometro ha una risoluzione di mezzo grado centigrado e incrementa di uno per ogni 0,5 °C. In altre parole, per ottenere il corretto valore di temperatura, basta dividere per due il byte letto in prima lettura, ovvero effettuare uno shift del dato di un posto a destra.

LISTATO 1

```
rem ********************************
rem *          LISTATO 1           *
rem *  Termometro per temperature  *
rem *          positive            *
rem *    di Giovanni Di Maria      *
rem ********************************

program termo01

' Lcd module connections
dim LCD_RS as sbit at RB4_bit
    LCD_EN as sbit at RB5_bit
    LCD_D4 as sbit at RB0_bit
    LCD_D5 as sbit at RB1_bit
    LCD_D6 as sbit at RB2_bit
    LCD_D7 as sbit at RB3_bit
    LCD_RS_Direction as sbit at TRISB4_bit
    LCD_EN_Direction as sbit at TRISB5_bit
    LCD_D4_Direction as sbit at TRISB0_bit
    LCD_D5_Direction as sbit at TRISB1_bit
    LCD_D6_Direction as sbit at TRISB2_bit
    LCD_D7_Direction as sbit at TRISB3_bit
' End Lcd module connections

dim temperatura1_letta as byte
dim temperatura2_letta as byte
dim temperatura_reale as float
dim centinaia as byte
dim decine as byte
dim unita as byte
dim decimi as byte
dim temp as word

main:

ANSEL  = 0        ' Configure AN pins as dig-
ital I/O
ANSELH = 0

Lcd_Init()
Lcd_Cmd(_LCD_CLEAR)
Lcd_Cmd(_LCD_CURSOR_OFF)
Lcd_Out(1, 1, "Temperatura:")

while (TRUE)
    '–- Perform temperature reading
    Ow_Reset(PORTE, 2)       ' Reset
    Ow_Write(PORTE, 2, 0xCC) ' Skip lettura
ROM
    Ow_Write(PORTE, 2, 0x44) ' Inizia con-
versione
    Delay_ms(200)
    Ow_Reset(PORTE, 2)       ' Reset
    Ow_Write(PORTE, 2, 0xCC) ' Skip lettura
ROM
    Ow_Write(PORTE, 2, 0xBE) ' Inizia
ricezione dati a 9 bit
    Delay_ms(200)

    rem —- Prima Lettura—————-
    temperatura1_letta = Ow_Read(PORTE, 2) '
Prima lettura LSB (temperatura)
    rem —- Seconda Lettura—————-
    temperatura2_letta = Ow_Read(PORTE, 2) '
Seconda lettura MSB

    Lcd_Out(2, 1, "                ")  ' 16
spazi
    rem –- Processa e visualizza le cifre del-
la temperatura
    if temperatura2_letta=0 then
           temperatura_reale = temperatu-
ra1_letta / 2.0
        temp=temperatura_reale*10
        centinaia=temp / 1000
        Lcd_chr(2, 1, centinaia+48)
        temp=temp - ( centinaia*1000 )
        decine=temp / 100
        Lcd_chr(2, 2, decine+48)
        temp=temp - ( decine*100 )
        unita=temp / 10
        Lcd_chr(2, 3, unita+48)
        Lcd_chr(2, 4, ",")
        temp=temp - ( unita*10 )
        decimi=temp
        Lcd_chr(2, 5, decimi+48)
        lcd_chr(2,7,223)
        lcd_chr(2,8,"C")
    else
        Lcd_Out(2, 1, "Sotto ZERO")
    end if
    Delay_ms(1000)
  wend
end.
```

TABELLA 1: LETTURA DI ALCUNI VALORI DI TEMPERATURA POSITIVA.

	PRIMA LETTURA			SECONDA LETTURA
TEMPERATURA °C	VALORE LETTO DECIMALE	VALORE LETTO BINARIO	VALORE LETTO ESADECIMALE	VALORE LETTO BINARIO
0,0	0	00000000	0000	00000000
0,5	1	00000001	0001	00000000
1,0	2	00000010	0002	00000000
1,5	3	00000011	0003	00000000
2,0	4	00000100	0004	00000000
2,5	5	00000101	0005	00000000
3,0	6	00000110	0006	00000000
3,5	7	00000111	0007	00000000
4,0	8	00001000	0008	00000000
4,5	9	00001001	0009	00000000
5,0	10	00001010	000A	00000000
10,0	20	00010100	0014	00000000
15,0	30	00011110	001E	00000000
20,0	40	00101000	0028	00000000
30,0	60	00111100	003C	00000000
40,0	80	01010000	0050	00000000
50,0	100	01100100	0064	00000000
70,0	140	10001100	008C	00000000
100,0	200	11001000	00C8	00000000
110,0	220	11011100	00DC	00000000
120,0	240	11110000	00F0	00000000
120,5	241	11110001	00F1	00000000
121,0	242	11110010	00F2	00000000
121,5	243	11110011	00F3	00000000
124,5	249	11111001	00F9	00000000
125,0	250	11111010	00FA	00000000

PRIMA ESPERIENZA: TERMOMETRO PER TEMPERATURE SOPRA LO ZERO

Per il primo esperimento costruiamo un semplice termometro digitale, con risoluzione ovviamente di 0,5 °C, che misuri la temperatura in un intervallo compreso tra 0 °C e 125 °C. I risultati sono visualizzati su un display LCD, collegato al microcontrollore. Il termometro è utile solo per misurare le temperature positive, ossia quelle comprese tra 0 °C e 125 °C. In caso di ambiente freddissimo (sotto lo zero), sarà invece visualizzata la frase "Sotto ZERO".

Schema elettrico

Il prototipo utilizza un microcontrollore PIC 16F887. Il quarzo utilizzato deve avere, come detto sopra, una frequenza minima di 4 Mhz, al fine di rilevare correttamente le temperature. Il quarzo deve essere, come al solito, coadiuvato dalle capacità di 22 pF. Il piedino di reset deve avere un potenziale alto, grazie alla resistenza di pull-up da 4,7k (o più). Il display LCD è connesso totalmente alla porta PORTB e usa solamente quattro delle otto linee dati. Il termometro DS1820 è connesso alla linea della porta RE2, a cui il firmware si riferisce. È importante prevedere la resistenza di pull-up da 4,7k su di essa, pena il suo non funzionamento

Il firmware

Il listato per implementare il termometro è relativamente semplice. Occorre comprendere bene la filosofia della comunicazione 1-Wire ma, soprattutto, occorre conoscere e capire i comandi supportati dal termometro DS1820. Allo scopo, si consiglia di leggere con molta attenzione il datasheet del dispositivo.
Il programma inizia con la definizione del funzionamento del display LCD, interamente collegato sulla PORTB. Seguono le dichiarazioni delle variabili. Dopo l'inizializzazione del display, inizia il ciclo vero e proprio, nel quale è rilevata e visualizzata la temperatura.

LISTATO 2

```
rem ********************************
rem *          LISTATO 2           *
rem *          Termostato          *
rem *    di Giovanni Di Maria      *
rem ********************************

program termo01

dim temperatura1_letta as byte
dim temperatura2_letta as byte
dim temperatura_reale as float

main:

ANSEL  = 0        ' Configure AN pins as dig-
ital I/O
ANSELH = 0

trisc=0
portc=0

while (TRUE)
    '–- Perform temperature reading
    Ow_Reset(PORTE, 2)       ' Reset
     Ow_Write(PORTE, 2, 0xCC) ' Skip lettura
ROM
     Ow_Write(PORTE, 2, 0x44) ' Inizia con-
versione
    Delay_ms(200)

    Ow_Reset(PORTE, 2)       ' Reset
     Ow_Write(PORTE, 2, 0xCC) ' Skip lettura
ROM
       Ow_Write(PORTE, 2, 0xBE) ' Inizia
ricezione dati a 9 bit
    Delay_ms(200)

    rem —-- Prima Lettura—————-
    temperatura1_letta =  Ow_Read(PORTE, 2) '
Prima lettura LSB (temperatura)
    rem —-- Seconda Lettura—————-
    temperatura2_letta =  Ow_Read(PORTE, 2) '
Seconda lettura MSB

    if temperatura2_letta=0 then
           temperatura_reale =  temperatu-
ra1_letta / 2.0
        rem —- Controlla temperatura per ter-
mostato—-
        if temperatura_reale >= 25 then
           portc.0=1
        else
           portc.0=0
        end if
    end if
    Delay_ms(1000)
  wend
end.
```

Essa è ricevuta per mezzo di comandi inviati al termometro. Leggendo il listato tale procedura è abbastanza semplice da comprendere. La preparazione delle informazioni da inviare al display LCD è un poco più complessa. Essa infatti si preoccupa di attingere e ricavare le singole cifre dalla temperatura (centinaia, decine, unità e decimi) e di visualizzarle singolarmente sul display, assieme al simbolo dei gradi centigradi (°C). Il motivo per cui la temperatura viene moltiplicata per dieci è quello di eliminare i difetti delle variabili floating point.

Segue in **tabella 2** un esempio di conversione, ipotizzando che la temperatura reale (variabile: temperatura_reale) sia di 103,5 gradi centigradi. Esso è particolarmente utile per capire i passaggi che determinano la visualizzazione su display delle varie cifre.

TABELLA 2: SIMULAZIONE DI CONVERSIONE TEMPERATURA/DISPLAY LCD.

STATEMENT DEL FIRMWARE MIKROBASIC	VALORE DELLA VARIABILE
temperatura_reale = temperatura1_letta / 2.0	103,5
temp = temperatura_reale*10	1035
centinaia = temp / 1000	1
temp = temp - (centinaia*1000)	35
decine = temp / 100	0
temp = temp - (decine*100)	35
unita = temp / 10	3
temp = temp - (unita*10)	5
decimi = temp	5

SECONDA ESPERIENZA: UN TERMOSTATO CHE ATTIVA UN CARICO

Il termostato è un interruttore comandato dalla temperatura. Esempi di termostato sono quelli contenuti in uno scaldabagno. Se l'acqua raggiunge una determinata temperatura, la resistenza si spegne. Un termostato può essere normalmente aperto o normalmente chiuso. Dipende dalle applicazioni di cui si ha necessità.

Vediamo come implementare un termostato con l'ausilio di un microcontrollore PIC e il termometro DS1820. Il listato è più semplice del precedente, in quanto non prevede alcuna unità di visualizzazione, ma solo un componente che attiva (o meno) un determinato carico. Occorre solo programmare il punto termico di attivazione (o di disattivazione), ossia la soglia sopra la quale il carico deve essere messo in opera o no.

Schema elettrico

Lo schema elettrico è molto semplice, in quanto non prevede il display LCD. Al suo posto si utilizza un transistor e un relè collegato al carico.
Anche questo prototipo utilizza un microcontrollore PIC 16F887. Il quarzo utilizzato deve avere, come detto sopra, una frequenza minima di 4 Mhz, al fine di rilevare correttamente le temperature. Il quarzo è coadiuvato dalle capacità di 22 pF.
Il piedino di reset deve avere un potenziale alto, grazie alla resistenza di pull-up da 4,7k (o più). Anche in questo caso il termometro DS1820 è connesso alla linea della porta RE2, a cui il firmware si riferisce.
È molto importante la presenza della resistenza di pull-up da 4,7k su di essa. La porta PORTC.0 comanda il carico, attraverso il transistor, configurato come interruttore elettronico.
Se lo stato logico di RC0 è alto, esso porta in saturazione il transistor, attivando il relè.
Il diodo D1, collegato in antiparallelo, serve per proteggere il transistor dalle extratensioni transitorie della bobina induttiva del relè.
Quest'ultimo è alimentato a 12 V, mentre la logica è a 5 V.

Il Firmware

Il **listato 2** è abbastanza semplice ed è simile al primo.
La temperatura reale è contenuta nella omonima variabile che, controllata ciclicamente ogni secondo, attiva o meno la porta PORTC.0.
In caso di misurazioni di temperature negative (sotto lo zero) occorre modificare il firmware, riferendosi alle tabelle di corrispondenza opportune, tenendo anche

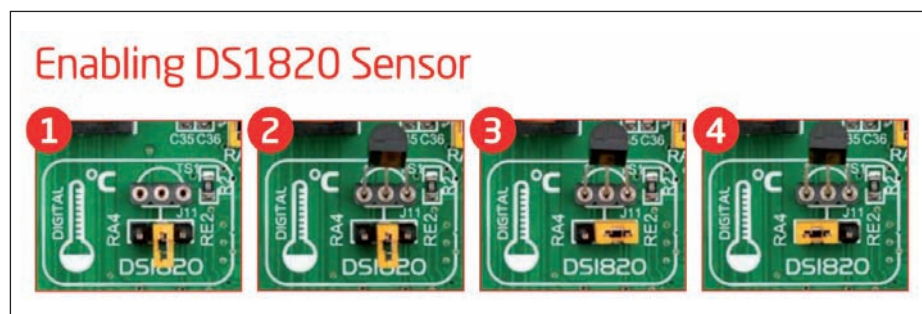



Figura 5: connessione del DS1820 sulla EasyPIC.

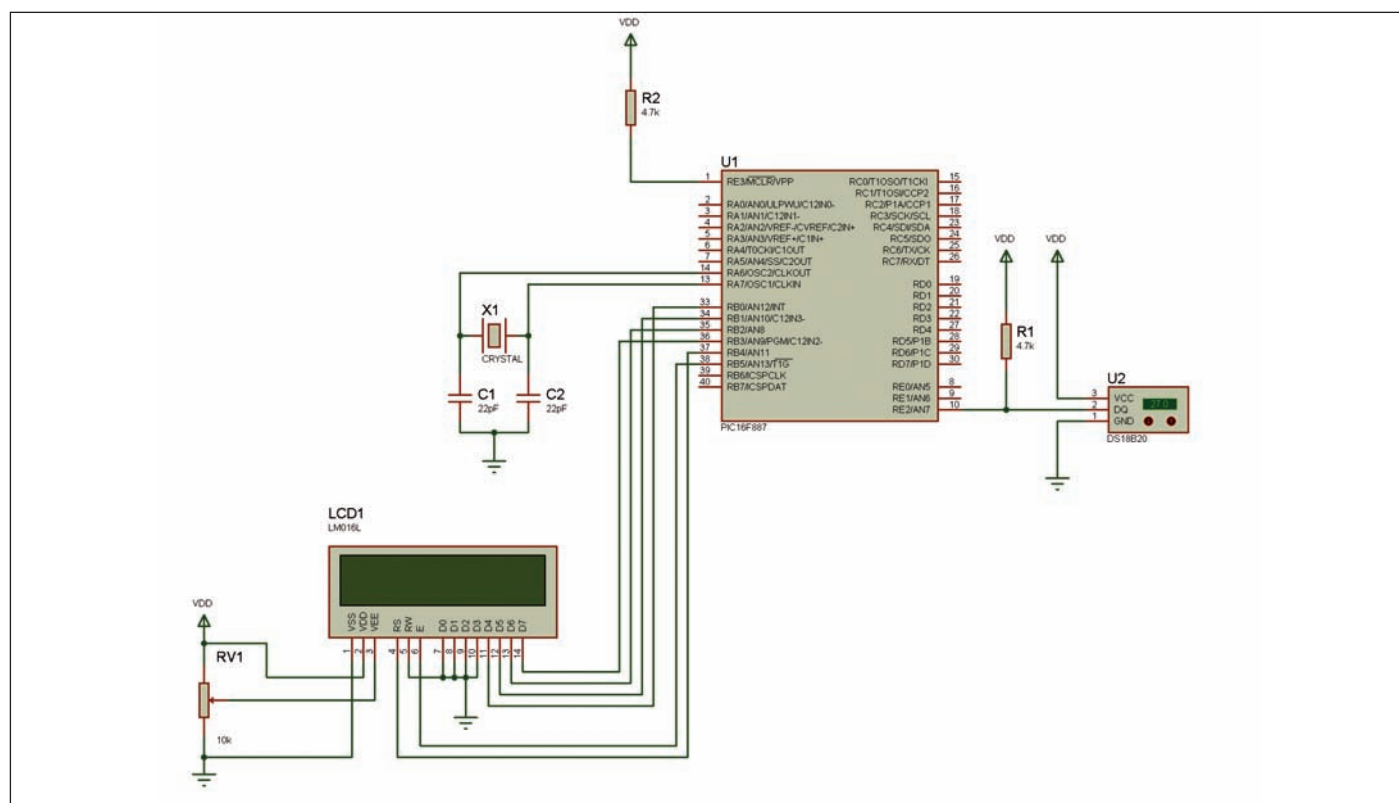



Figura 6: schema elettrico del termometro per temperature positive.

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	00000000 11111010	00FA
+25°C	00000000 00110010	0032h
$+^1/_2$°C	00000000 00000001	0001h
+0°C	00000000 00000000	0000h
$-^1/_2$°C	11111111 11111111	FFFFh
–25°C	11111111 11001110	FFCEh
–55°C	11111111 10010010	FF92h

Figura 7: relazione tra temperature e valori restituiti dal DS1820.

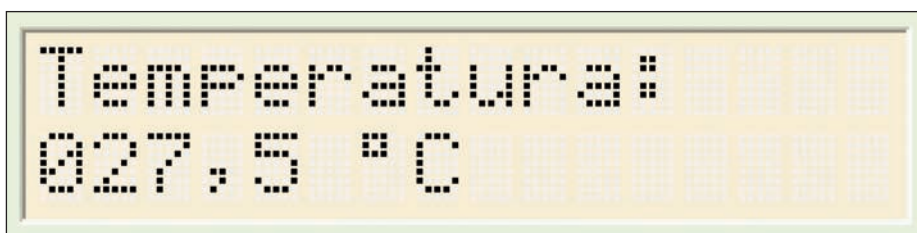



Figura 8: la temperatura visualizzata sul display LCD.

conto che la seconda lettura del DS1820 deve restituire un valore di 255 ($FF). L'esempio visto serve ad attivare un carico se la temperatura raggiunge i 25 gradi. Se essa si abbassa sotto tale limite, il carico viene disattivato. È possibile utilizzare un semplice diodo led (con relativa resistenza di limitazione) al posto del transistor e del relè, in fase di test, giusto per controllare l'alternanza degli stati logici della porta RC0. Per far alzare la temperatura è sufficiente "alitare" un po' direttamente sul termometro DS1820. Per modificare il valore termico di soglia è sufficiente agire sulla condizione IF presente nel ciclo ripetitivo. Per i puristi è opportuno prevedere e implementare una "isteresi" digitale, onde evitare veloci commutazioni in caso di temperatura fissa sul valore di soglia. In ogni caso il prototipo di esempio funziona egregiamente.

CONCLUSIONI

Siamo arrivati alla conclusione di questo corso avanzato sui microcontrollori. I lettori hanno molto apprezzato il lavoro svolto, fatto testimoniato anche dalle numerose lettere ed email ricevute. Con le varie lezioni abbiamo aggiunto tanti tasselli importanti alla conoscenza della programmazione dei microcontrollori con il mikroBasic. ❑

CODICE MIP 2831461

Figura 9: schema elettrico del termostato.

di ADRIANO GANDOLFO

Demo Board PER RASPBERRY PI

Nei precedenti numeri della rivista è stata presentata la scheda Raspberry. Come abbiamo visto, è dotata di un connettore che permette di dialogare direttamente con i chip e i moduli dei sottosistemi. In questo connettore sono presenti, oltre ai pin GPIO (General Purpose I/O), altri pin per accedere tramite interfacce SPI, I2C, UART seriale, e quelli di alimentazione a 3,3 V e 5 V. In quest'articolo vedremo come realizzare demo board con cui poterci esercitare pilotando le uscite o acquisendo gli ingressi. Nei prossimi numeri saranno analizzati i segnali presenti sul connettore della scheda Raspberry Pi e saranno presentati alcuni programmi per l'utilizzo della scheda demo.

Nei precedenti numeri è già stata ampiamente presentata la scheda Raspberry Pi, per chi avesse perso i numeri ecco un breve sunto.

LA SCHEDA RASPBERRY PI MODELLO B

La scheda visibile in **figura 1** misura 85,60 x 53,98 x 17 mm, il suo peso è di 45 g, presenta qualche sporgenza dovuta alla scheda SD e connettori che sporgono oltre i bordi.
Il modello visibile è quello più completo della scheda Raspberry Pi, il modello B: dotato di due porte USB ed Ethernet LAN 10/100, con un SoC Broadcom BCM2835 (che comprende un core ARM1176JZF-S a 700 MHz, un acceleratore grafico 3D Videocore IV e un decoder hardware capace di gestire video Full-HD 1080p a un bitrate massimo di 40Mbps).
Nella **tabella 1** ne sono riassunte le caratteristiche.

CONNETTORI PRESENTI SULLA SCHEDA

La scheda Raspberry Pi, come un normale PC, ha vari connettori **(figura 2)**: connettore di alimentazione **[1]**, una doppia USB **[2]**, il connettore per la connessione alla rete Ethernet **[3]**, una porta HDMI, un connettore RCA e un connettore tipo cuffie per l'uscita Audio/video **[4-5-6].**
In più, sono presenti interfacce a basso livello, utilizzabili per dialogare direttamente con i chip e moduli dei sottosistemi. Quello che utilizzeremo, è siglato P1 **[7]**.
Altri connettori sono: un CSI (Camera Serial Interface) e un DSI (Display Serial Interface), oltre a un collegamento seriale all'interno del connettore HDMI chiamato CEC (Consumer Electronics Control).

General Purpose Input/Output (GPIO)

Il connettore General Purpose **(figura 3)** di Input / Output è un gruppo di pin che possono essere configurati come pin d'ingresso o di uscita ed essere controllati (programmati) tramite software. In questo modo il Raspberry Pi permette a periferiche e schede di espansione di accedere alla CPU.
Il connettore denominato P1 presente sulla scheda Raspberry Pi è formato da 26 pin disposti su due file con un passo di 2,54 millimetri.
Sono presenti pin specifici per l'accesso alle interfacce I²C, SPI, UART, sono poi presenti i pin di alimentazione 3,3 V, 5 V e GND.
I livelli di tensione della GPIO sono a 3,3 V e non tollerano 5 V **(figura 4)**.
Nell'interfacciarsi con la scheda, occorre prestare attenzione poiché non c'è protezione contro la sovratensione.
Tutti i pin GPIO possono essere riconfigurati per fornire funzioni alternative come SPI, PWM, I²C e così via, come visibile nella **tabella 2**.
La massima corrente che può essere assorbita dal pin di alimentazione 3,3 V è 50 mA, mentre quello del pin dell'alimentazione +5 V dipende dagli altri assorbimenti, ma si aggira intorno a 300-500 mA.

DEMO BOARD PER SCHEDA RASPBERRY PI

La Demo Board visibile nella **figura 5** permetterà di compiere esperimenti d'interfacciamento con la scheda Raspberry: il collegamento con quest'ultima è effettuato con un cavo piatto a 26 poli che termina, a entrambi i lati, con un connettore 13+13 pin.
Il connettore della scheda Raspberry che

TABELLA 1

SoC (System on a Chip)	**Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)**
CPU	**700 MHz ARM1176JZF-S core (famiglia ARM11)[2]**
GPU:	**Broadcom VideoCore IV,OpenGL ES 2.0, 1080p30 H.264 high-profile decode**
Memoria (SDRAM):	**256 Megabytes (condivisa con la GPU)**
Porte USB 2.0:	**2 (attraverso un hub USB integrato)**
Output video:	**Connettore RCA per il video composito, HDMI**
Output audio:	**3,5 mm jack, HDMI**
Memoria:	**SD / MMC / SDIO card slot**
Collegamenti di rete:	**Ethernet 10/100 (RJ-45)**
Periferiche di basso livello:	**2x13 header pins per GPIO, SPI, I²C, UART, +3,3 volt, +5 volt**
Real-time clock:	**Nessun orologio o batteria**
Potenza assorbita:	**700 mA, (3,5 watt)**
Alimentazione:	**5 Volt via MicroUSB o GPIO header**
Dimensioni:	**85,60 × 53,98 mm**
Sistemi operativi supportati:	**Debian GNU/Linux, Fedora, Arch Linux e Gentoo**

Tabella 1: caratteristiche scheda Raspberry Pi.

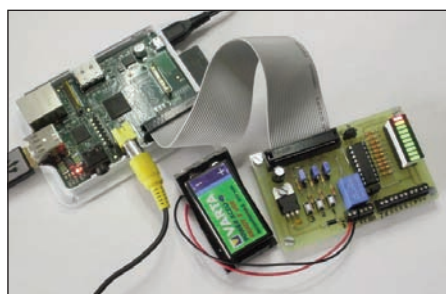



Figura 1: scheda Raspberry Pi.

Figura 2: posizione connettori su Scheda Raspberry Pi.

si utilizzerà è denominato P1 **(figura 6)** e su di esso sono presenti, oltre ai pin GPIO (General Purpose I/O), alcuni pin per accedere tramite interfacce SPI, I²C, UART seriale oltre a quelli di alimentazione.
La nostra scheda demo, utilizzerà solamente gli otto pin I/O, i due pin di alimentazione a 3,3 V e 5 V e il pin di massa.
La scheda possiede:
- connettore verso Raspberry PI;
- morsettiere per connessione: alimentazione esterna, alle 8 uscite, al relè;
- circuito di alimentazione esterna con batteria tramite regolatore **LM7805**;
- interfaccia di uscita delle 8 porte I/O tramite **ULN2803A**;
- barra LED per segnalazione attivazione uscite, presenza delle tensioni di alimentazione;
- tre pulsanti connessi alle porte I/O;
- un relè.

È possibile vedere un breve filmato illustrativo presente su YouTube al link http://youtu.be/iBiV6llgQXk.

DESCRIZIONE DEL CIRCUITO

Passiamo adesso alla descrizione del circuito della Demo Board visibile in **figura 7**, mentre nella **tabella 3** è riportato l'elenco dei componenti utilizzati.
La scheda può essere vista secondo le singole sezioni:
- sezione Connettore;
- sezione alimentazione;
- sezione interfaccia I/O;
- sezione uscita relè.

SEZIONE CONNETTORE

Il connettore **(figura 8)** permette il collegamento tra la scheda Raspberry Pi e la Demo Board.
I pin di I/O collegati sono solamente 8, come riportato nella **figura 9:** di questi, 3 hanno la doppia funzione di Input e Output mentre gli altri sono solamente di Output. Sono poi presenti i tre pin dell'alimentazione: +3,3 V, +5 V e GND.

Sezione alimentazione

La sezione dell'alimentazione **(Figure 10 e 11)** è incentrata sul noto integrato LM7805 **(figura 12)**, un regolatore di tensione fissa con uscita di +5 V, 1 A.
La tensione in ingresso è fornita tramite il connettore X1, dopo il quale si trova il diodo D1 che protegge il circuito da un'eventuale inversione di polarità.
Sono poi presenti alcuni condensatori di filtraggio, il diodo LED BL1-10 segnala la presenza dell'alimentazione.
Il Jumper J4 permette di scegliere la fonte di alimentazione che può essere esterna o derivata dalla scheda Raspberry Pi.
L'accensione del diodo BL1-9 segnala quando la scheda è connessa alla Raspberry Pi e che questa è alimentata.

Sezione interfaccia I/O

La sezione d'interfaccia **(figura 13)** si occupa di interfacciare i segnali da e per la scheda Raspberry Pi.
L'integrato processore SoC (System-on-a-Chip) della Broadcom utilizzato sulla scheda Raspberry Pi è basato su tecnologia CMOS e pertanto i pin di uscita operano su livelli logici 3,3 V.
Se questo può andar bene per l'interfacciamento con altri integrati di tipo

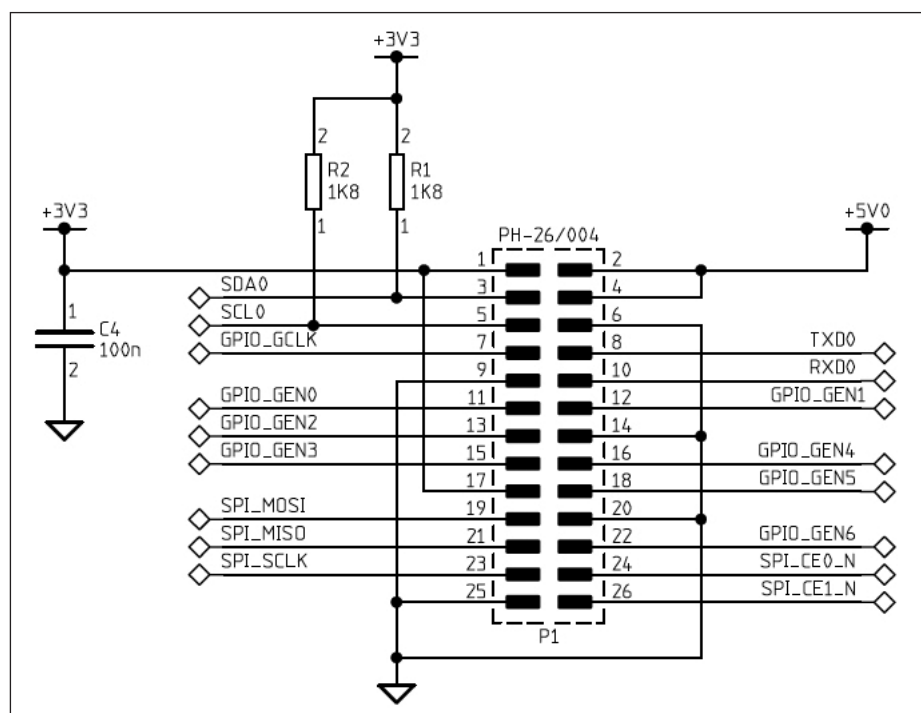



Figura 3: dettaglio dello schema della scheda Raspberry Pi.

CMOS è molto meglio lavorare con integrati di tipo TTL che operano a livelli logici 5 V.
Serve quindi un circuito per convertire i segnali digitali CMOS (3,3 V) a livelli logici TTL (5 V). L'integrato scelto il tipo ULN2803A **(figura 14)**, è molto versatile e con parametri di funzionamento ideali per la nostra applicazione.
Questo integrato contiene al suo interno 8 circuiti Darlington **(figura 15**) è può quindi convertire le 8 linee CMOS del Raspberry Pi a livelli logici TTL. 5 V
Le uscite di questo integrato sono chiamate "open collector" per cui quando i Darlington sono attivati, permettono il passaggio della corrente verso il pin di massa. L'integrato può essere alimentato sia dalla stessa Raspberry Pi che dal regolatore presente sulla scheda.
L'alimentazione è fornita tramite i due pin: 10 per i +5 V, 9 per GND, ed è possibile pilotare carichi sino a 500 mA.
Nel caso di alimentazione tramite la Raspberry Pi, si ricorda che è presente un fusibile SMD da 750 mA che scatterà se si preleverà più corrente dalla presa di alimentazione.
Le uscite dell'integrato sono connesse sia ai LED presenti nella barra a LED, sia alle morsettiere presenti sulla scheda.
Il modello della barra LED scelto **(figura 18)** è prodotto da ARKLED, riporta la sigla SB1K1Y8G e ha la particolarità di possedere, a differenza dei tipi standard in cui tutti i dieci sono dello stesso colore, 8 LED di colore verde, uno di colore arancio e uno rosso.
In questo modo si è potuto differenziare le

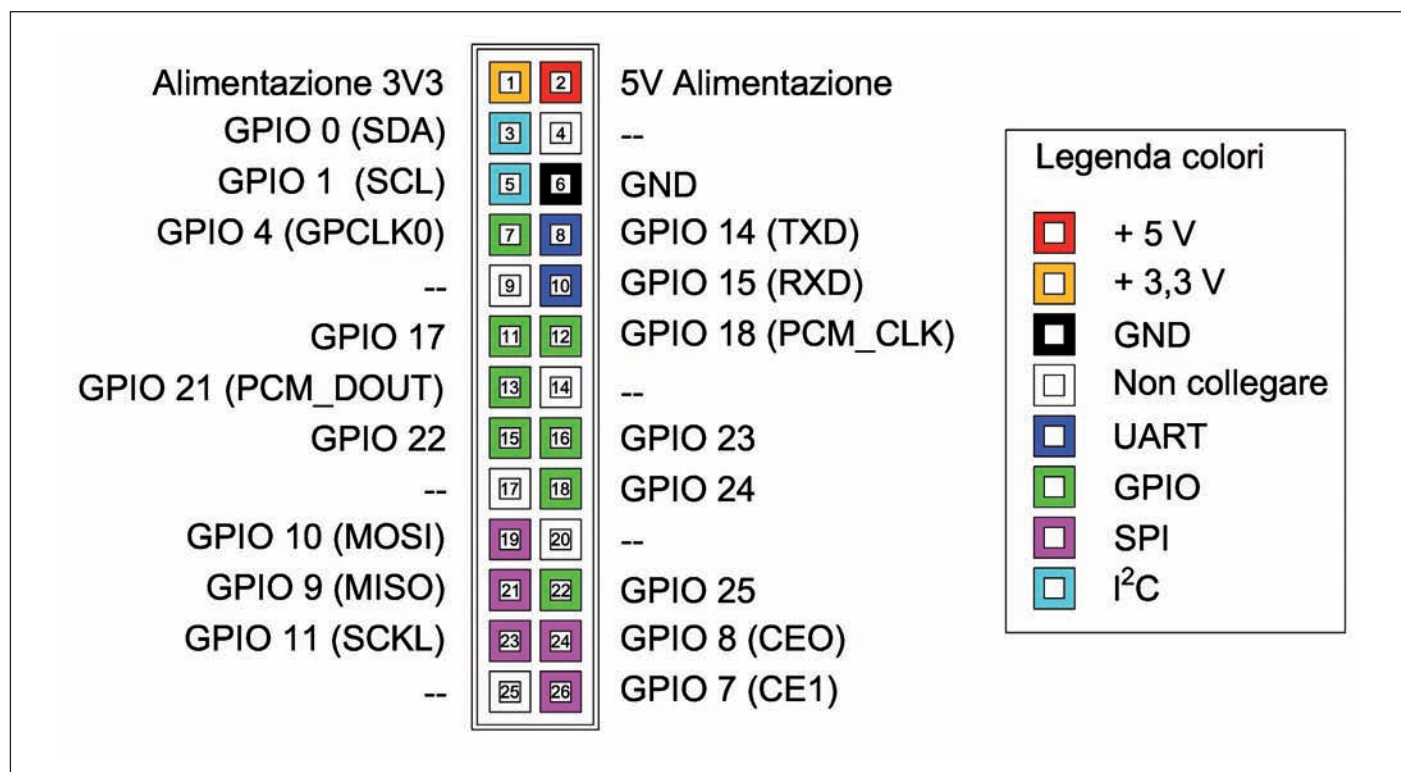



Figura 4: funzione dei pin del connettore P1.

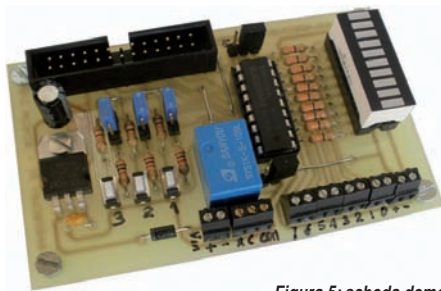



Figura 5: scheda demo per Raspberry Pi.

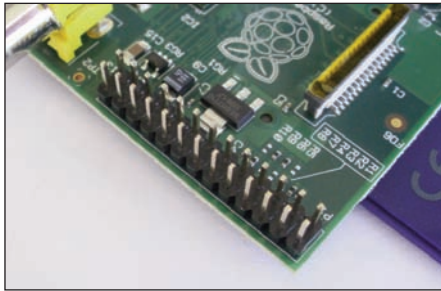



Figura 6: connettore P1 su Scheda Raspberry Pi.

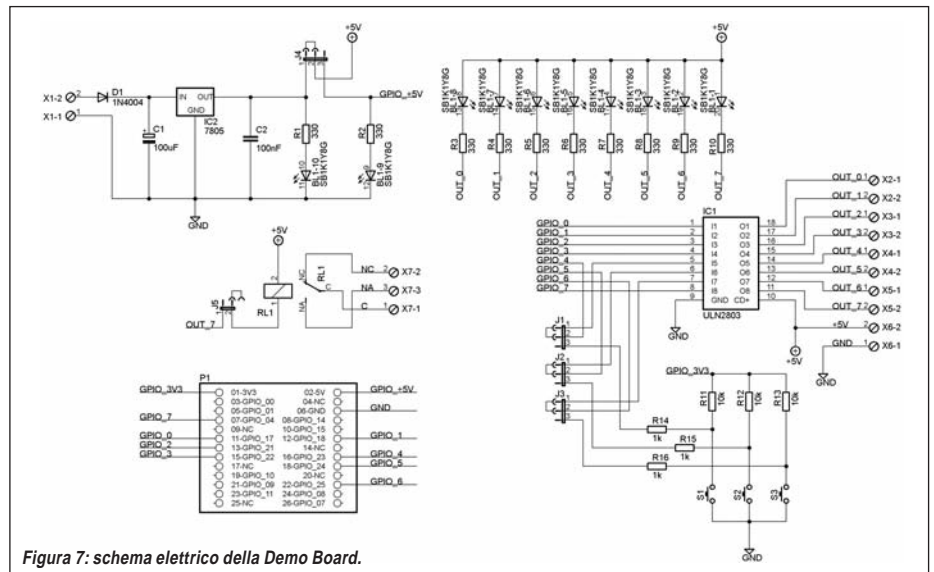



Figura 7: schema elettrico della Demo Board.

TABELLA 2

NUMERO PIN	NOME DEL PIN	NOTE HARDWARE	FUNZIONE ALTERNATIVA	ALTRE FUNZIONI ALTERNATIVE
P1-02	5V0	Fornita tramite fusibile		
P1-04	DNC	Non connettere		
P1-06	GND			
P1-08	GPIO 14	Boot to Alt 0 ->	UART0_TXD	ALT5 = UART1_TXD
P1-10	GPIO 15	Boot to Alt 0 ->	UART0_RXD	ALT5 = UART1_RXD
P1-12	GPIO 18			ALT4 SPI1_CE0_N ALT5 = PWM0
P1-14	DNC	Non connettere		
P1-16	GPIO23			ALT3 = SD1_CMD ALT4 = ARM_RTCK
P1-18	GPIO24			ALT3 = SD1_DATA0 ALT4 = ARM_TDO
P1-20	DNC	Non connettere		
P1-22	GPIO25			ALT4 = ARM_TCK
P1-24	GPIO08		SPI0_CE0_N	
P1-26	GPIO07		SPI0_CE1_N	
P1-01	3.3 V	50 mA max current draw		
P1-03	GPIO 0	Resistenza 1K8 pull up	I2C0_SDA	I2C0_SDA
P1-05	GPIO 1	Resistenza 1K8 pull up	I2C0_SCL	I2C0_SCL
P1-07	GPIO 4			GPCLK0
P1-09	DNC	Non connettere		
P1-11	GPIO17			ALT3 = UART0_RTS, ALT5 = UART1_RTS
P1-13	GPIO21		PCM_DIN	ALT5 = GPCLK1
P1-15	GPIO22			ALT3 = SD1_CLK ALT4 = ARM_TRST
P1-17	DNC	Non connettere		
P1-19	GPIO10		SPI0_MOSI	
P1-21	GPIO9		SPI0_MISO	
P1-23	GPIO11		SPI0_SCLK	
P1-25	DNC	Non connettere		

Tabella 2: funzioni primarie e alternative dei pin connettore P1.

varie segnalazioni come riportato nella **tabella 4.**
Resta inteso che è possibile utilizzare un modello di barra LED standard. Attualmente ne esistono con LED di vari colori: rossi, verdi, blu.
Ogni LED è protetto da una resistenza limitatrice, il suo valore è stato calcolato considerando la tensione di alimentazione del LED di 5 V, una caduta di tensione che per il colore rosso è di 1,8 V e una corrente che attraversa il LED di 10 mA, da cui il valore della resistenza di 330 ohm.
Oltre alle uscite sono presenti 3 pulsanti NA utilizzati come INGRESSI. Questi possono essere connessi alle porte di I/O della Raspberry Pi tramite tre Jumper J1-J2-J3 che li sconnettono dalle porte di IC1.
Poiché le porte GPIO della Raspberry Pi non sono protette con un buffer, sono presenti resistenze (R14-15-16) da un 1 k tra l'interruttore e il pin GPIO come precauzione per evitare danni se la porta GPIO fosse accidentalmente impostata come uscita.
I tre interruttori sono collegati alla linea GND in modo che, quando sono premuti, il circuito è completato tra il GPIO pin e GND.
Completano il circuito le tre resistenze di pull-up (R11-12-13) che presentano un valore di 10 k'Ω.

TABELLA 3:

C1	100 nF multistrato
C2	10 F elettrolitico verticale
R1..R10	330 Ω ¼ W R11,12,13 10 kΩ ¼ W
R13,14,15 1	kΩ ¼ W
D1	1N4004
BL1	Display LED 10 grafico tipo SB1K1Y8G ARKLED+ zoccolo 20 pin SIL
IC1	ULN2803 + zoccolo 18 pi SIL
IC2	LM 7805
P1	Connettore maschio per connettori a perforazione d'isolante 26 poli
J1,2,3,4	Pinhead maschio 1x3 + ponticello
J5	Pinhead maschio 1x2 + ponticello
RL1	Relè 5 V tipo SYSY1K-S-105L o equivalente
S1,2,3	Minipulsante 2 pin da circuito stampato (basso profilo)
X1,2,3,4,5,6,	Morsettiera due poli a vite passo 2,5 mm
X7	Morsettiera tre poli a vite passo 2,5 mm

Tabella 3: elenco componenti scheda Demo Board.

TABELLA 4:

COLORE LED	FUNZIONE
Led Rosso	Alimentazione +5 V da regolatore IC2
Led Arancio	Alimentazione da Raspberry Pi
Led verde	Attivazione uscita porta IC1

Tabella 4: funzione di segnalazione della barra a LED.

TABELLA 5:

JUMPER	FIGURA	FUNZIONE
J1	28-28	Abilitazione Pulsante S1
J2	28-29	Abilitazione Pulsante S2
J3	28-29	Abilitazione Pulsante S3
J4	30	Scelta della fonte di alimentazione
J5	31-32	Abilitazione relè

Tabella 5: funzione dei jumper.

Sezione relè

Sulla scheda **(figura 19)** è presente un piccolo relè tipo SYSY1K-S-105L **(figura 21)** che può essere attivato inserendo il jumper J5 **(figura 20)**, in questo modo l'uscita 7 dell'integrato IC1 sarà collegata alla bobina e il relè potrà essere attivato.
I contatti del relè: Comune, Normale Aperto e Normale Chiuso sono presenti sulla morsettiera X7 della scheda.

COSTRUZIONE DELLA SCHEDA

Per la realizzazione della scheda è previsto l'uso di un circuito stampato. Si dovrà quindi scaricare, dal sito di Fare Elettronica, il PDF che riporta la traccia in scala 1:1. Esiste anche la possibilità di scaricare dal sito i file per il programma EAGLE con cui ognuno potrà adattare il circuito per i propri utilizzi.
Per la sua realizzazione si utilizzerà una basetta in vetronite (monofaccia) di dimensioni 89x55 mm circa **(figura 22)**. Il metodo potrà essere quello della fotoincisione o del trasferimento termico utilizzando i cosiddetti fogli blu (PRESS-N-PELL).
Una volta inciso il rame si verificherà, in controluce o mediante l'utilizzo di un multimetro, che non vi siano cortocircuiti soprattutto tra le piste più vicine (zona connettore). Si passerà quindi alla foratura della stessa, utilizzando:

- punta da 0,8 mm per i fori delle resistenze, condensatori, zoccoli intergrato e barra LED, ponticelli, relè;
- punta da 1 mm per connettori, morsettiere, integrato regolatore, diodo, jumper;

Figura 8: connettore Pi.

Figura 9: pin di I/O utilizzati dalla Demo Board.

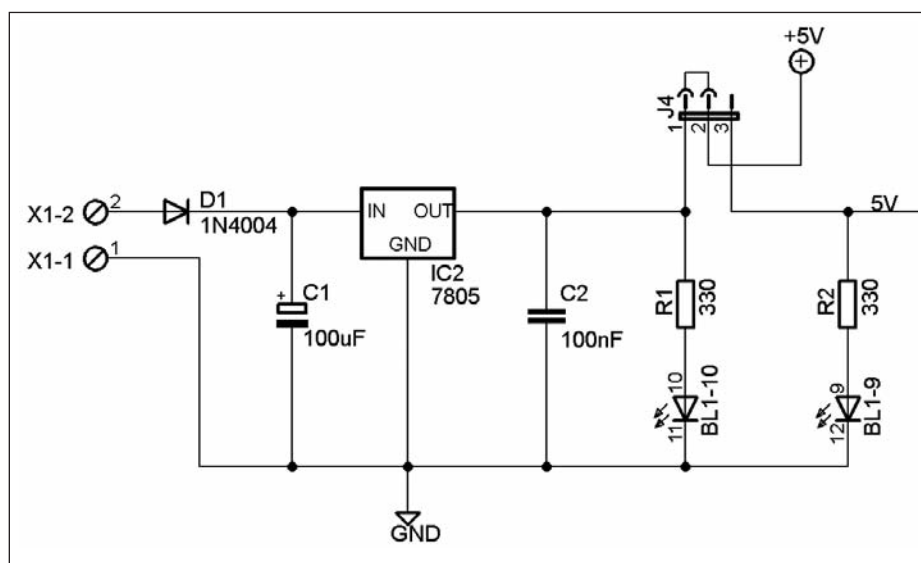



Figura 10: schema elettrico della sezione alimentazione della Demo Board.

- punta da 3,5 mm per fori di fissaggio stampato e integrato regolatore.

Quindi si disporranno e salderanno i componenti seguendo lo schema riportato nella **figura 23**.

Per la saldatura si utilizzerà un piccolo saldatore a punta fine, della potenza di circa 25-30 W. Si inizierà dai componenti a più basso profilo, quindi dai ponticelli, le resistenze, il diodo D1 rispettandone la polarità. Si passerà ai due zoccoli, uno per l'integrato IC1 e uno per la barra LED; quindi si salderanno i tre pulsanti miniatura.

Per il posizionamento del regolatore IC2 occorrerà preventivamente piegarne i terminali, aiutandosi con un paio di pinze. Il fissaggio sarà poi completato tramite vite e dado M3 utilizzando l'apposito foro. Si salderà successivamente il connettore Pi rispettando la posizione del pin 1, evidenziato da un triangolo sul corpo del connettore. si passa poi alla saldatura delle morsettiere, i Jumper, e i condensatori, rispettando la polarità per quanto riguarda C1. Per ultimo si dovrà saldare il relè. Terminato il montaggio, si possono montare IC1 e la barra LED rispettando l'orientamento.

Collaudo della scheda

Terminato il montaggio, e prima di collegarlo alla scheda Raspberry, è necessario procedere a un attento controllo, visivo o meglio mediante un multimetro, verificando che non siano presenti cortocircuiti tra le piste.

REALIZZAZIONE DEL CAVO

Per il collegamento della scheda Raspberry Pi con la Demo Board, occorre realizzare un cavo di collegamento. I componenti necessari sono: 2 connettori femmina volanti flat-cable a 26 poli e uno spezzone di cavo piatto flat cable 26 poli, di circa 20-30 cm (**figura 25**).

La sua realizzazione non è complessa e l'unico attrezzo necessario è rappresentato da una morsa da banco che permette di "chiudere" in modo parallelo le due parti che formano il connettore, facendo in modo che i pin possano perforare la guaina del cavo.

Nel montaggio occorre prestare attenzione affinché il connettore più esterno del cavo colorato di rosso sia posto in corrispondenza del simbolo a forma di triangolo presente sul connettore **(figura 26)**, che corrisponde al pin 1. A lavoro finito il cavo dovrà i presentarsi come in **figura 27**.

Descrizione jumper della scheda

Prima del collegamento della scheda, occorre predisporre i vari jumper presenti sulla scheda stessa.

A questo punto si potrà collegare la Demo Board con la Raspberry Pi, come visibile in **figura 29**.

Figura 11: sezione di alimentazione.

Figura 12: piedinatura dell'integrato LM7805.

Figura 13: schema elettrico della sezione alimentazione della Demo Board.

Figura 14: piedinatura dell'integrato ULN2803A.

INFORMAZIONI SUL SISTEMA OPERATIVO DELLE RASPBERRY

La scheda Raspberry Pi, come un normale computer, ha bisogno di un sistema operativo, abbreviato in OS (Operating System), che è un insieme di componenti software le quali garantiscono l'operatività di base, coordinando e gestendo le risorse hardware di processamento e memorizzazione, le periferiche, le risorse software e facendo da interfaccia con l'utente. Senza di esso, quindi, non sarebbe possibile l'utilizzo della Raspberry Pi e di altri software più specifici, come applicazioni o librerie software.
Come già detto, l'OS della Raspberry Pi risiede su una memoria SD da inserire nell'apposito slot presente.
Di questo OS esistono varie versioni, quello utilizzato per le prove è denominato Raspbian, un sistema operativo debian-based che nasce appositamente per essere utilizzato su Raspberry Pi.
Raspbian è un'ottimizzazione di Debian per lavorare con il processore di Raspberry Pi e si è dimostrato più veloce rispetto ad altre versioni.
È possibile scaricarlo a questo link: http://www.raspberrypi.org/downloads.

IL LINGUAGGIO DI PROGRAMMAZIONE PYTHON

Per la scrittura del programma di gestione, personalmente ho scelto il linguaggio di programmazione Python, che è di tipo ad alto livello, orientato agli oggetti e adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.
Altri linguaggi cui Python è spesso paragonato sono Perl, Java, JavaScript, Visual Basic.
Il linguaggio fu ideato da Guido Van Rossum all'inizio degli anni Novanta. Il nome fu scelto per via della passione dell'ideatore per il gruppo Monty Python.
Python è un linguaggio pseudocompilato: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione .py) e, se sintatticamente corretto, di eseguirlo. In Python non esiste una fase di compilazione separata (come avviene in C, per esempio) che generi un file eseguibile partendo dal sorgente.
L'essere pseudointerpretato rende Python un linguaggio portabile. Una volta scritto un sorgente, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme attualmente utilizzate, siano esse di casa Apple (Mac) o PC (Microsoft Windows e GNU/Linux). Semplicemente, basta la presenza della versione corretta dell'interprete.
Un'altra particolarità è che Python è free software: non solo il download dell'interprete per la propria piattaforma, ma anche l'utilizzo nelle proprie applicazioni.
Queste caratteristiche hanno fatto di questo programma il protagonista di un enorme diffusione in tutto il mondo. Questo perché garantisce lo sviluppo rapido (e divertente) di applicazioni di qualsiasi complessità in tutti i contesti: dal desktop al web, passando dallo sviluppo di videogiochi allo scripting.
Per quanto riguarda l'interprete, questo è

Figura 15: circuito interno di uno degli 8 Darlington dell'integrato ULN2803A.

Figura 16: integrato IC1.

Figura 17: barra LED.

Figura 18: dimensioni e piedinatura della barra LED.

Figura 19: sezione relè.

Figura 20: schema elettrico della sezione relè.

Figura 21: schema elettrico della sezione relè.

già installato all'interno del sistema Operativo Raspbian, sarà sufficiente fare un doppio clic sull'icona posta sul desktop, denominata IDLE 3. In questo modo sarà caricato il programma interprete che si presenta come un'editor dei testi che ci permetterà di scrivere, salvare, caricare i programmi il cui file avrà come estensione "py".

INSTALLAZIONE DELLE LIBRERIE RPI.GPIO PER PYTHON

Per la gestione della Demo Board è possibile utilizzare una specifica libreria Python chiamata **RPi.GPIO** che consente di configurare in modo semplice i pin GPIO sia in lettura sia in scrittura all'interno di uno script Python.
Per utilizzarle è necessario installare il pacchetto poiché non è attualmente incluso nell'immagine del SO.
Questo è disponibile presso http://pypi.python.org/pypi/RPi.GPIO: attualmente la versione più aggiornata è la 0.4.1a.
Se esiste una versione più aggiornata sarà necessario apportare le modifiche appropriate per il numero di versione nei comandi sotto riportati.
Per installarlo accertarsi che la scheda Raspberry sia connessa a Internet, quindi lanciare una linea di comando (cioè LXTerminal) e digitare i seguenti comandi.

Fase 1 - scaricare la libreria:
http://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.4.1a.tar.gz
wget

Fase 2 - estrarre l'archivio in una nuova cartella:
tar zxf RPi.GPIO-0.4.1a.tar.gz

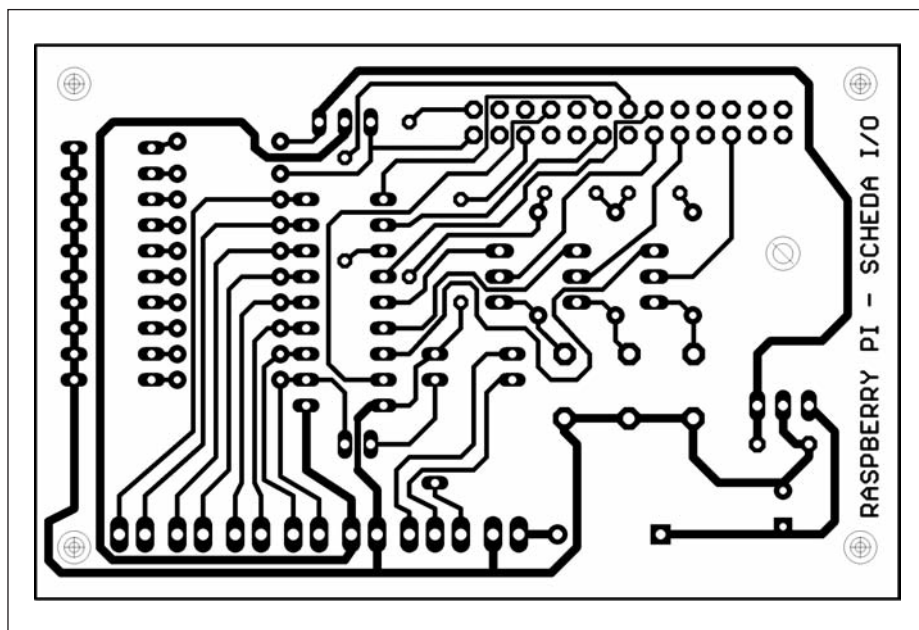



Figura 22: circuito stampato.

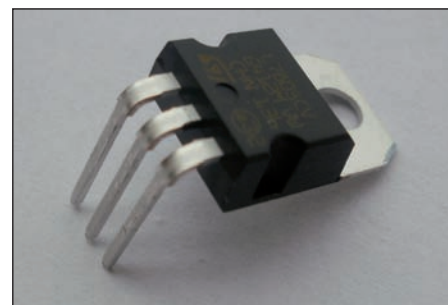



Figura 24: integrato regolatore IC2.

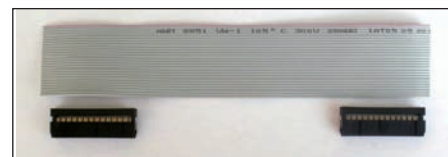



Figura 25: componenti cavo di collegamento.

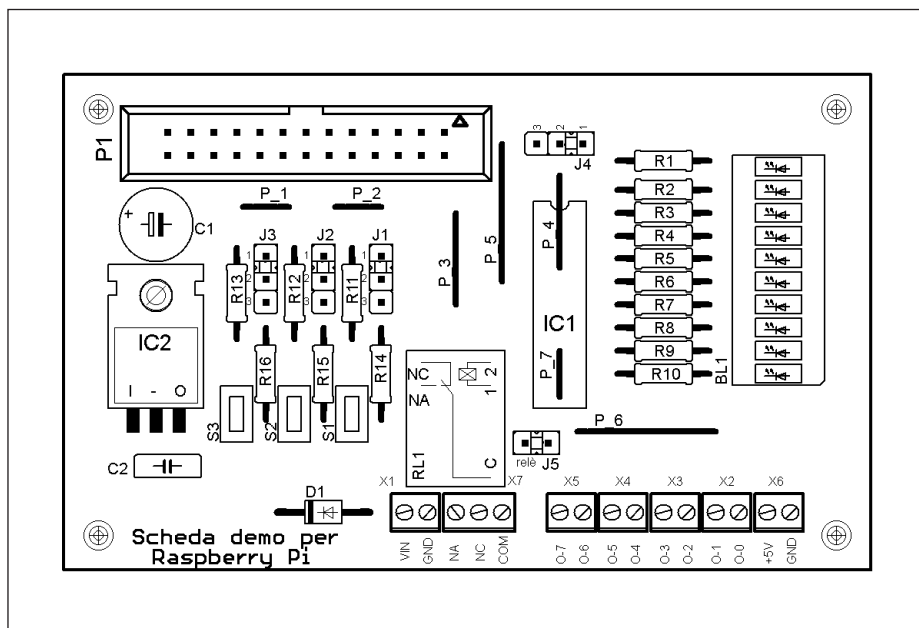



Figura 23: piano di montaggio dei componenti.

Figura 26: posizione della tacca di riferimento.

Figura 27: componenti cavo di collegamento.

Fase 3 - individuare la nuova directory:

```
cd RPi.GPIO-0.4.1a
```

Fase 4 - installare la libreria:

```
sudo python setup.py install
```

Fase 5 - rimuovere la directory e il suo contenuto:
cd ..

```
sudo rm-rf RPi.GPIO-0.4.1a /
```

Fase 6 - eliminare il file di archivio:

```
rm-RPi.GPIO 0.4.1a.tar.gz
```

Si potrà ora utilizzare la libreria in Python.

PROGRAMMA DI PROVA

Per testare la scheda si può usare il programma il cui listato è scaricabile dal sito di *Fare Elettronica*. Questo farà lampeggiare tutti gli otto LED connessi alle porte.

Il programma dovrà essere copiato utilizzando la Shell del programma Python oppure scaricato da sito della rivista e copiato nella SD del SO.
Il funzionamento del programma è molto semplice. La prima linea:

```
#!/usr/bin/python
```

riporta la posizione del programma

Figura 28: pulsanti disabilitati.

Figura 29: collegamento delle schede.

Figura 30: memoria SD con OS.

```
*Test_scheda.py - /home/pi/Test_scheda.py*
File Edit Format Run Options Windows Help
#!/usr/bin/python

import time
import RPi.GPIO as GPIO

# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)

# set up the GPIO channels
GPIO.setup(11, GPIO.OUT) # GPIO 0
GPIO.setup(12, GPIO.OUT) # GPIO 1
GPIO.setup(13, GPIO.OUT) # GPIO 2
GPIO.setup(15, GPIO.OUT) # GPIO 3
GPIO.setup(16, GPIO.OUT) # GPIO 4
GPIO.setup(18, GPIO.OUT) # GPIO 5
GPIO.setup(22, GPIO.OUT) # GPIO 6
GPIO.setup(7, GPIO.OUT) # GPIO 7

GPIO.output(11, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(12, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(13, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(15, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(16, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(18, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(22, True) #GPIO a 1 (HIGH), Led acceso
GPIO.output(7, True) #GPIO a 1 (HIGH), Led acceso
Ln: 27 Col: 49
```

Figura 32: programma di test.

Python. Le successive:

```
import time
import RPi.GPIO as GPIO
```

servono a importare le due librerie: la prima, **time**, è una libreria standard usata per la temporizzazione, mentre la **RPi.GPIO** va istallata secondo le indicazioni del precedente paragrafo.
Segue l'impostazione di come chiameremo i pin, che può essere:

```
GPIO.setmode(GPIO.BOARD)
```

oppure

```
GPIO.setmode(GPIO.BCM)
```

Vengono poi impostate le varie porte, tutte come uscite, e creato un ciclo che pone ogni uscita a true, accendendo il LED. Quindi, dopo una pausa di un secondo stabilita dall'istruzione:

```
time.sleep(1).
```

Una volta scritto il programma e salvato per esempio come Test_scheda.py, si potrà mandare in esecuzione mediate il comando RUN, oppure dalla riga di comando

```
sudo python Test_scheda.py
```

È possibile vedere il funzionamento del programma nel filmato presente su YouTube al link http://youtu.be/0n4MzWPjj9g.

CONCLUSIONI

Abbiamo terminato il montaggio della scheda e verificato il suo funzionamento. Potremmo ora realizzare altri programmi per gestire i pulsanti, il relè, o altri dispositivi da collegare alle morsettiere d'uscita.
Nei prossimi numeri saranno illustrati altri programmi e applicazioni che potremo testare con la nostra scheda appena costruita. ❏

CODICE MIP 2831658

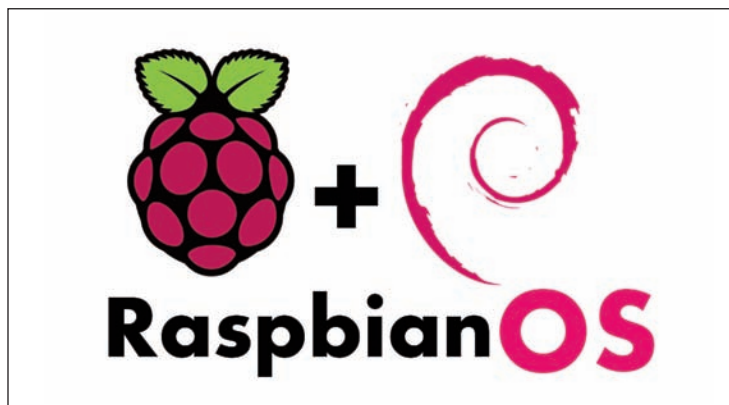



Figura 31: logo del sistema operativo Raspbian.

PER approfondire...

- http://www.raspberrypi.org/ Sito ufficiale della fondazione che ha creato la scheda Raspberry.
- http://www.raspberrypi.org/downloads Per scaricare il sistema operativo della Raspberry Pi.
- http://downloads.element14.com/raspberryPi1.html?isRedirect=true Link sito Farnel per ordinare la scheda Rasberry Pi.
- http://it.rs-online.com/web/generalDisplay.html?id=raspberrypi Link sito RS per ordinare la scheda Raspberry Pi.
- http://www.python.it/ sito ufficiale della comunità italiana per il linguaggio di programmazione Python.
- http://www.raspberrypi.org/downloads Per scaricare il sistema operativo della RaspBerry Pi.
- http://pypi.python.org/pypi/RPi.GPIO Per scaricare la libreria per gestire le porte I/O.

Interfacciamento dei processori *(parte tredicesima)*

Display a matrice CON DECODER BINARI

La gestione delle matrici di LED è decisamente accattivante ma, nella sua forma nativa, non è possibile attivarla tramite la porta parallela di un PC e comunque, anche se operata con un microcontrollore, ne intacca sensibilmente le risorse; in questa puntata studieremo la soluzione di questo problema.

Nella puntata precedente abbiamo visto che l'interfacciamento diretto dei display a matrice di punti richiede al dispositivo di controllo come minimo dodici linee d'uscita, per garantire le quali sono necessarie almeno due porte; la struttura di questi dispositivi optoelettronici divide i LED interni in gruppi composti solo da una piccola parte di essi, collegandone insieme gli anodi in modo da costituire una "Riga", e collegando poi tra loro i catodi dei LED che occupano la medesima posizione in ciascuna Riga, dando luogo ad un numero di "Colonne" pari a quello dei LED in essa contenuti.

Le configurazioni più diffuse sono quelle a matrice 7x5 e a matrice 8x8 e richiedono comunque un registro d'uscita a 8 bit per le 7 o 8 Righe e un altro registro d'uscita a 8 bit per le 5 o 8 Colonne: qualunque microcontrollore non avrà problemi (se non quello di consumare molte linee di I/O) per garantire (con le sue porte PortA e PortB) il servizio diretto a entrambe le strutture ma, di certo, esso

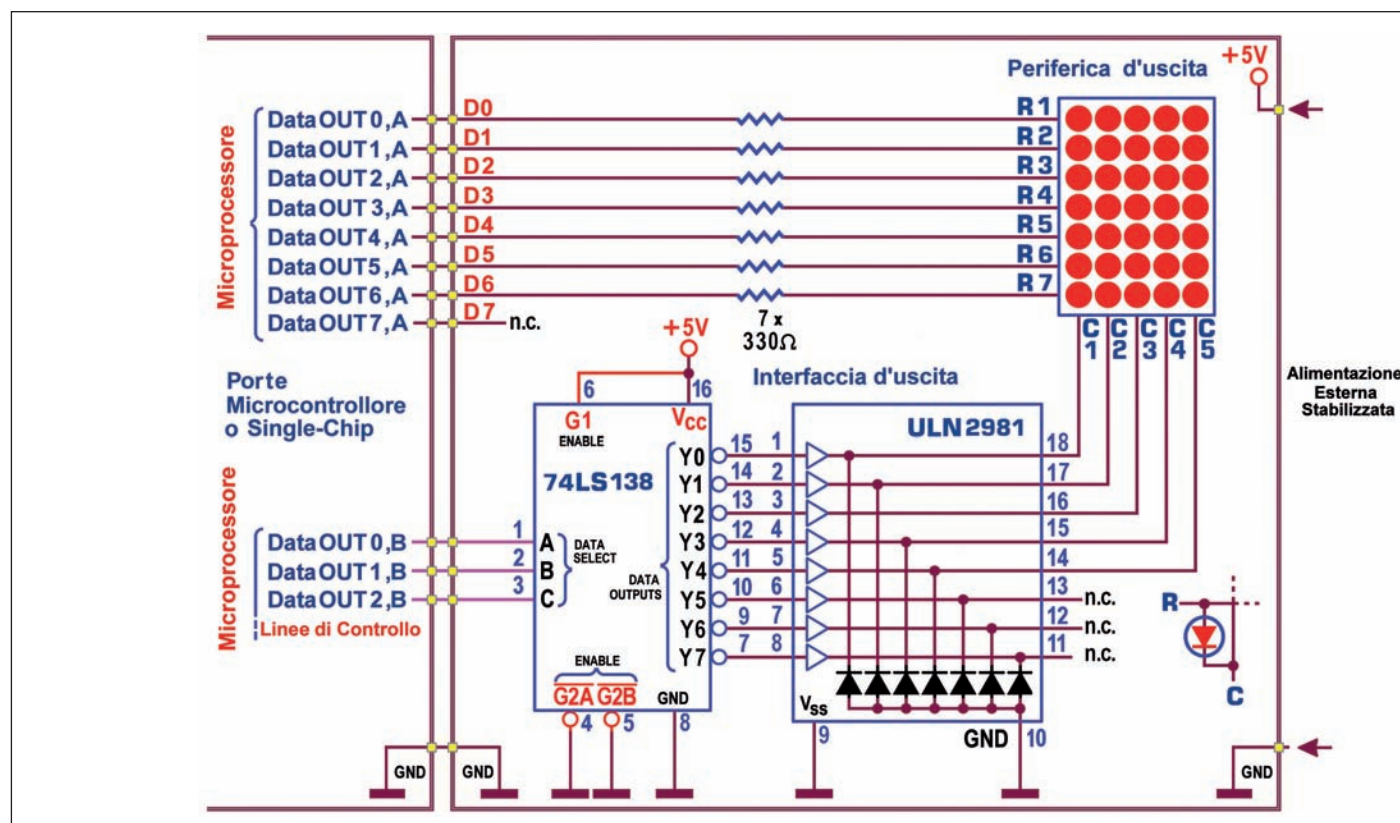



Figura 1: Dot Matrix 7x5 LED Display: controllo con microcontrollore tramite 3-Line to 8-Line Decoder/DMUX 74LS138.

di GIORGIO OBER

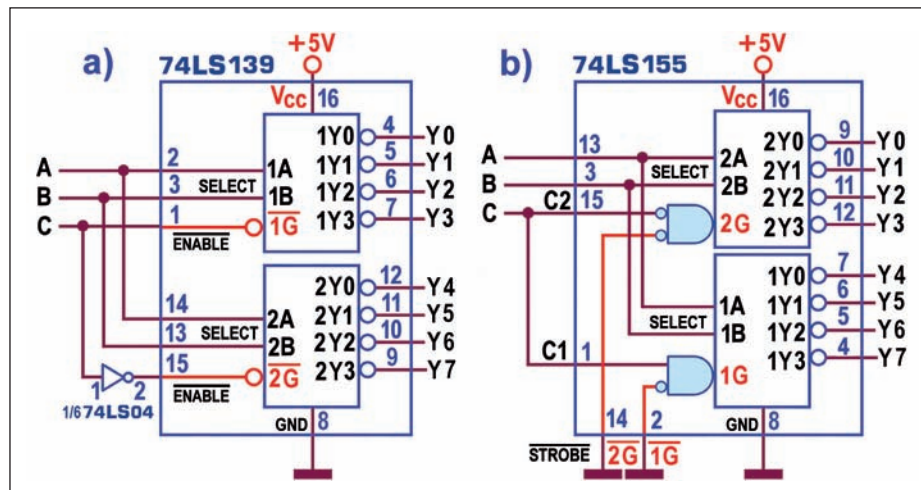
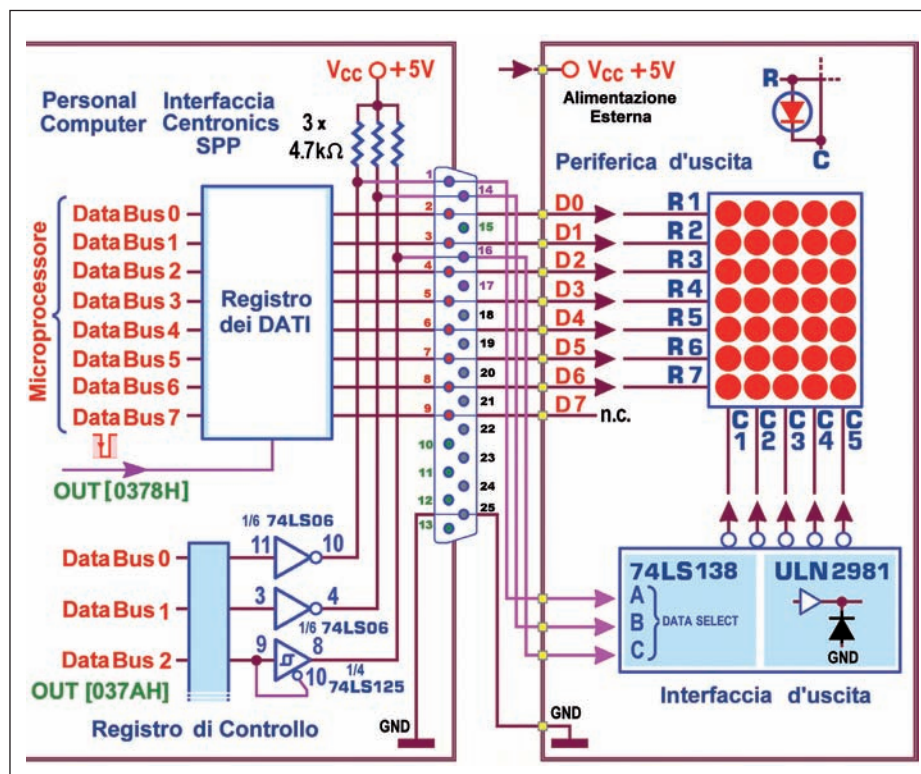



Figura 2: sostituire il 74LS138 con Dual 2-Line To 4-Line Decoder/DMUX: a) con74LS139; b) con 74LS155.

non potrà essere assolto dalla porta parallela SPP di un Personal Computer, per l'insufficiente numero di linee d'uscita su di essa disponibile.

Ricordando che, in ogni caso, è necessario ricorrere alla tecnica del multiplexing dell'informazione, per chiarezza, continueremo a riferirci ad una matrice 7x5 a catodo comune e a dividere i relativi 35 bit in cinque parti (una per Colonna), affidando in sequenza ciascuna di esse alla porta d'uscita associata alle sette Righe e attivando (cioè portando a massa) di volta in volta una sola Colonna; ma è evidente che la stessa tecnica è utilizzabile anche per una matrice 7x5 ad anodo comune, affidando l'informazione alle Colonne e attivando in sequenza le Righe, una dopo l'altra. Ferma restando l'ipotesi di affidare direttamente a una porta d'uscita a gestione delle 7 Righe, poiché ciascuna di esse sarà chiamata a erogare la corrente di un solo LED, possiamo ritenere che le sue linee d'uscita siano in grado di fornirla senza problemi, sia essa la porta PortA di un single-chip o il Registro d'uscita Dati 0378H/0278H della porta parallela del PC, con riferimento allo standard SPP (Standard Parallel Port).

Vediamo ora come sostituire le cinque linee d'uscita destinate al controllo delle 5 Colonne, di certo facilmente gestibili dalla PortB di un microcontrollore ma non dal secondo registro d'uscita di una porta parallela, ovvero dalle linee del Registro di Controllo 037AH/027AH.

La soluzione più logica per ridurre il numero di linee di controllo è quella di affidare la gestione delle Colonne a un decoder binario: se il visualizzatore è composto da un solo display a matrice di punti è sufficiente usare un decoder come il 74LS138 (un 3-Line To 8-Line Decoder/DMUX); si tratta di un dispositivo chiamato a interpretare (= decodificare) un codice binario puro a 3 bit, predisposto sulle sue 3 linee d'ingresso (Data Select), attivando poi la sola uscita (tra 8) il cui numero corrisponde alla parola ricevuta.

La **figura 1** mostra il circuito a esso relativo; notiamo la presenza di uno strato di bufferizzazione, necessario per il fatto che ogni Colonna può essere costretta a gestire l'accensione di più LED, anche tutti e sette, rendendo troppa la corrente da assorbire da parte di una logica TTL LS.

Il componente utilizzato è un ULN2981, un High-voltage/High-current Darlington Transistor Array contenente 8 high current sink driver, ciascuno basato su una coppia di transistor in configurazione Darlington che, come è noto, si comporta come un singolo transistor ad alto guadagno di corrente; in questo modo la corrente gestita da ogni linea può arrivare fino a 500 mA, con picchi di 600 mA, consentendo anche alle normali logiche, TTL o CMOS, di pilotare periferiche che richiedono alta corrente.

A differenza del più noto ULN2003 (che

Figura 3: Dot Matrix 7x5 LED Display: controllo con porta parallela SPP tramite 3-Line to 8-Line Decoder/DMUX 74LS138.

per altro contiene un driver Darlington in meno) il ULN2981 ha la caratteristica di avere uscite non invertenti, condizione indispensabile in questo caso, data la logica attiva bassa delle uscite del decoder che lo precede; in questo modo le condizioni necessarie alla tecnica di multiplexing sono assicurate tenendo alte tutte le Colonne, meno quella puntata dal codice binario fornito in uscita sui 3 bit della porta PortB del microcontrollore.
Ricordiamo che i driver di questa categoria dispongono di diodi polarizzati inversamente (clamp diodes) sia in ingresso sia in uscita, per assicurare un buon grado di sicurezza; le uscite di ciascun canale sono a collettore aperto e consentono (sebbene non sia il caso del nostro attuale progetto) di utilizzare per la periferica a esse collegata tensioni di alimentazione diverse, anche maggiori, fino a un massimo di 50 V, rispetto a quella (di tipo logico) usata per il controllo in ingresso, di solito di 4,2 V fino a un massimo di 5 V.
In aggiunta, le stesse uscite del ULN2981 possono essere facoltativamente collegate a massa attraverso un ulteriore diodo di soppressione, previsto internamente, per dissipare eventuali possibili picchi di tensione durante il controllo di eventuali carichi induttivi; per motivi di logica praticità gli anodi di tutti questi diodi opzionali interni sono collegati tra loro e resi disponibili su un piedino dedicato, pin10, detto GND.
Inutile sottolineare che la presenza di 8 canali indipendenti (decoder + transistor Darlington) rende questo progetto ideale anche per pilotare matrici di 8x8 LED; una possibile variante, funzionalmente identica, consiste nel sostituire il decoder “da 3 a 8” (74LS138) con un doppio decoder “da 2 a 4” (Dual 2-Line To 4-Line Decoder/DMUX) come il 74LS139 o il 74LS155; la **figura 2** ne sottolinea pregi e difetti:
a) con il 74LS139 la cosa è possibile rinunciando alla possibilità di controllarne l’abilitazione (Enable), ma per poter disporre della terza linea di selezione è necessario coinvolgere anche un operatore NOT (1/6 di 74LS04), rendendo la soluzione antieconomica;
b) l’uso del più moderno 74LS155 è la scelta migliore: le due metà hanno ora le 2 linee di selezione A e B in comune, richiedendo per questo solo 2 piedini, invece di 4; i due piedini resi liberi sono utilizzati per controllare le rispettive abilitazioni (Strobe) con una propria linea, C1 e C2; poiché le due linee di controllo sono attive in modo contrapposto (per la presenza di un inverter interno) è sufficiente unirle tra loro esternamente per ottenere il terzo ingresso di selezione C, trasformando il componente nel desiderato in decoder “da 3 a 8”.
In ogni caso la gestione software di questo progetto è molto semplice: per poter ricostruire l’informazione desiderata sulla matrice di LED (di solito un carattere alfanumerico o un simbolo grafico) bisogna in primo luogo predisporre una variabile contenente le 5 parole da trasferire sulle Righe, una per ciascuna delle 5 Colonne previste da questo visualizzatore; ciascuna parola avrà la dimensione di un byte e conterrà i bit necessari per tenere accesi (con 1) o spenti (con 0) i rispettivi LED della Colonna ad essa relativa; poiché le Righe previste sono sette, il valore del bit più significativo (bit7) di ciascuna parola è irrilevante e potrà essere tenuto a 0.
Dunque, sarà necessario predisporre il

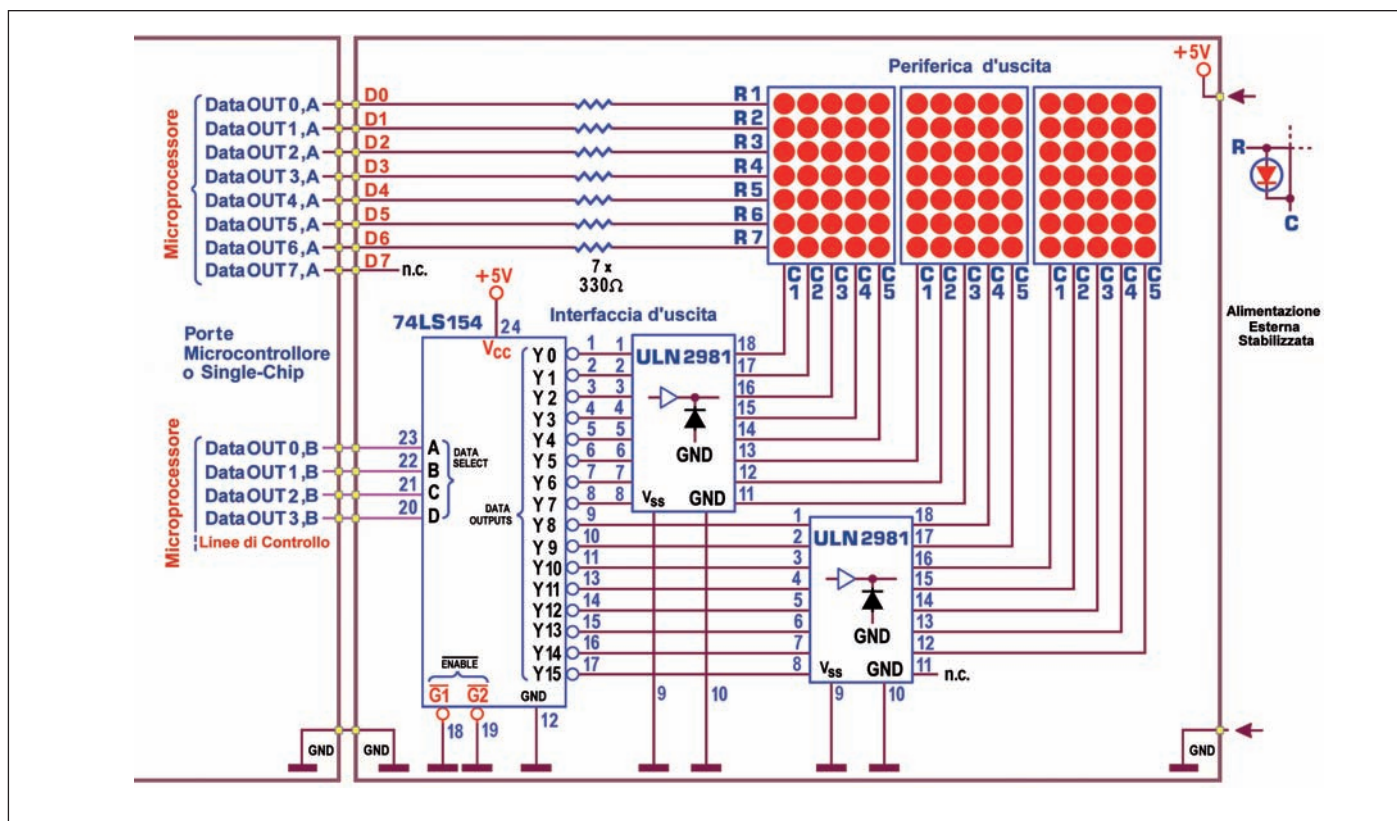



Figura 4: Dot Matrix 7x5 LED Display: controllo con microcontrollore tramite 4-Line to 16-Line Decoder/DMUX 74LS154.

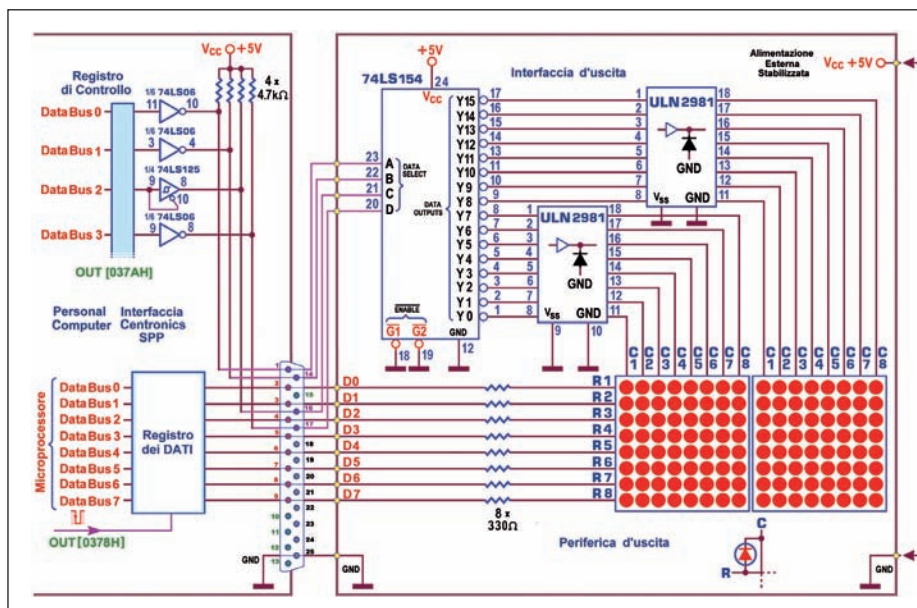



Figura 5: Dot Matrix 8x8 LED Display: controllo con porta parallela SPP tramite 4-Line to 16-Line Decoder/DMUX 74LS154.

primo byte sulla PortA e subito dopo spedire il codice 000 sulle tre linee meno significative della PortB, programmate in uscita: immediatamente le Righe che risulteranno forzate a 1 (dai bit con questo valore logico presenti sulla prima porta) accenderanno i corrispondenti LED, collegati "a catodo comune" sulla prima Colonna (quella a sinistra del display), messa a massa dall'uscita Y0 del decoder e bufferizzata dal canale non invertente del ULN2981 a esso collegato.

A questo punto il programma deve attendere un breve istante (dell'ordine del millisecondo) e poi provvedere alla stessa operazione con il secondo byte previsto dalla struttura del simbolo desiderato, scrivendolo sulla PortA e rendendone visibile il contenuto sulla seconda Colonna, forzata a massa dal codice 001 scritto subito dopo su PortB; dopo aver consumato un altro millisecondo la medesima operazione sarà condotta con il terzo byte (attivato forzando a 010 la PortB), e così via per il quarto (con PortB=011) e per il quinto byte (con PortB=100).

Dopo poco più di 5 ms tutte le Colonne sono state interessate dall'informazione necessaria per mostrare l'immagine "a punti" sul visualizzatore, probabilmente appena intuibile; ma solo ripetendo all'infinito tutti i comandi descritti essa sarà visibile stabilmente, per l'effetto di persistenza ottica; la commutazione continua tra le Colonne consentirà al simbolo di formarsi in modo sempre più nitido al crescere della frequenza di scansione, fino a risultare del tutto leggibile e privo di sfarfallamenti, non appena essa avrà raggiunto un valore intorno ai 50 Hz (pari ad una cadenza di 20 ms).

Il progetto è ora facilmente adattabile alla porta parallela di un Personal Computer, anche con riferimento al semplice standard SPP: nonostante sia sempre meno presente, essa è facilmente disponibile con l'aggiunta di un'economica scheda di I/O e rimane pur sempre un eccellente strumento per interfacciare numerosi dispositivi esterni con la potenza indiscutibile di un PC, di gran lunga superiore a quella di un single-chip.

La **figura 3** mostra lo schema delle parti interne della parallela necessarie per pilotare la parte destra della **figura 1**, cioè la periferica Dot Matrix 7x5 LED Display interfacciata con un 74LS138-/ULN2981.

Con questa tecnica, coinvolgendo nel progetto un 74LS154 (4-Line To 16-Line Decoder/DMUX), è possibile pilotare anche più di una matrice di punti; la PortB del microcontrollore dovrà ora rendere disponibili 4 linee d'uscita, per fornire il richiesto codice binario a 4 bit, da 0000 a 1111, necessario per la selezione delle 16

```
        CALL DeskTop

xxx00:  CALL KEYwait
        AND  AL,11011111B

        CMP  AL,"F"
        JNZ  xxx01
        LEA  SI,Tab_FE
        CALL Display
        JMP  xxx00

xxx01:  CMP  AL,"S"
        JNZ  xxx02
        LEA  SI,Tab_Tot
        MOV  CX,24
        CALL Scroll
        JMP  xxx00

xxx02:  CMP  AL,"Q"
        JNZ  xxx00

_OUT:   ...  ........
```

Prepara il Desktop
NO — Ci sono tasti premuti? — SI
NO — Premuto "F"? — SI
Gestione di Testo fisso
NO — Premuto "S"? — SI
Gestione di Testo scorrevole
NO — Vuoi terminare? — SI

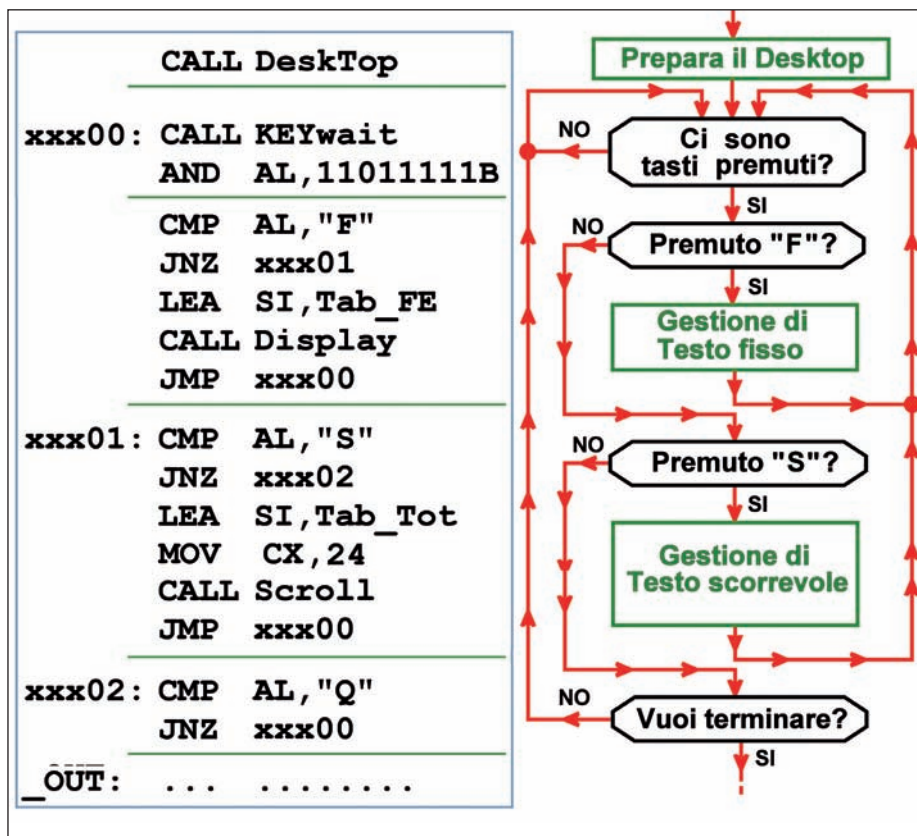



Figura 6: codice assembly per la gestione di due Dot Matrix 8x8 LED Display.

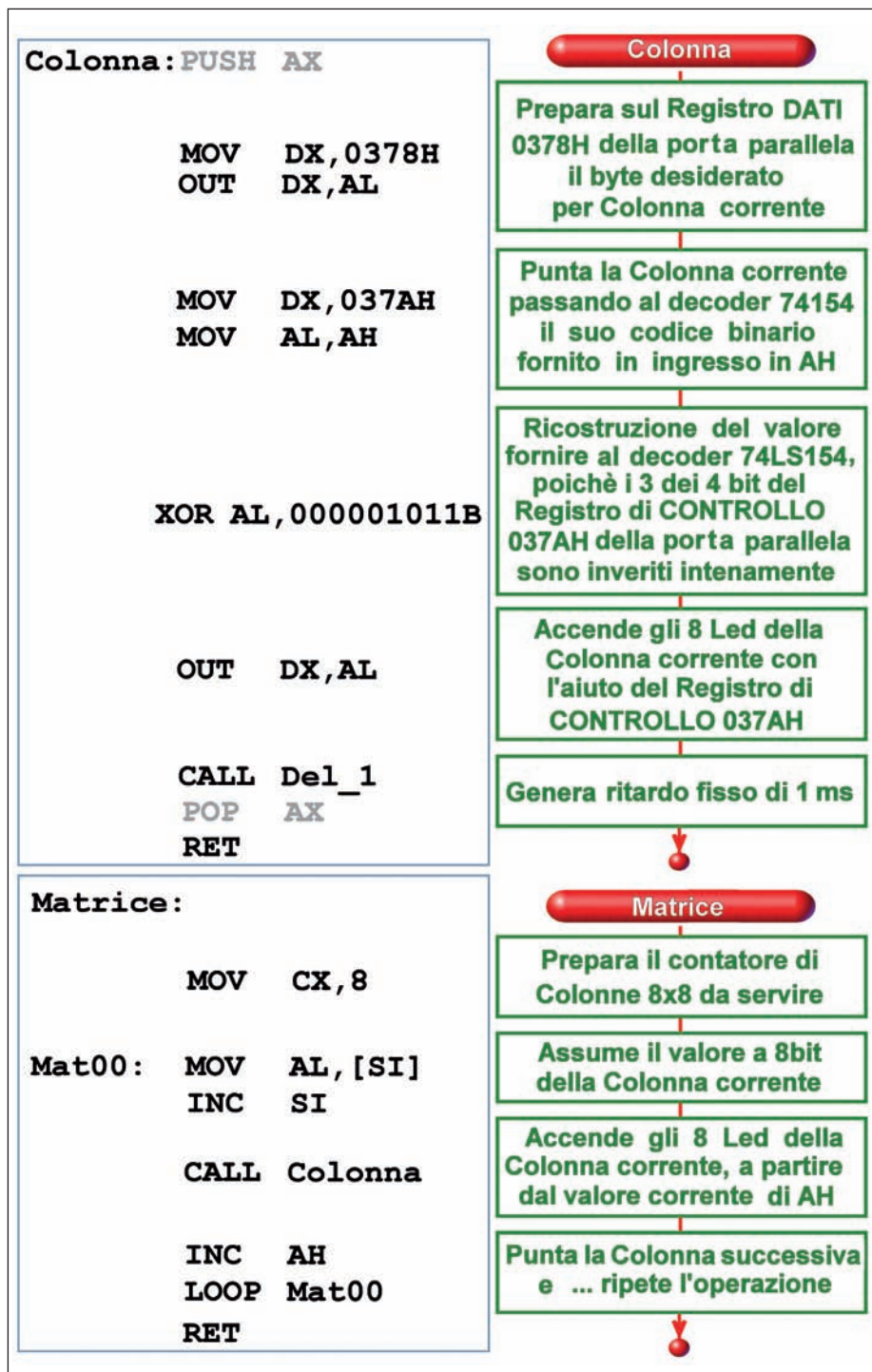



Figura 7: Dot Matrix 5x7 LED Display: procedure assembly per la gestione: A) di una singola Colonna; B) dell'intera Matrice.

uscite; la **figura 4** mostra lo schema adatto al controllo fino a 3 matrici 7x5.
In ogni caso è facile capire che, potendo selezionare in sequenza fino a 16 Colonne, l'interfaccia 74LS154/2xULN2981 è ideale anche per la gestione di due matrici 8x8. Questo nuovo progetto è perfettamente compatibile anche con la porta parallela di un PC, dato che il suo Registro di Controllo 037AH/027AH è in grado di garantire fino a 4 linee d'uscita. La **figura 5** mostra lo schema delle parti interne della parallela necessarie per pilotare due matrici di 8x8 LED, ma è utilizzabile anche con l'interfaccia proposta nella parte destra della **figura 4**, dedicata alla gestione di 3 matrici 7x5. Per collaudare questa struttura ho progettato un semplice sorgente in codice assembly 80x86, il cui Main Program è proposto in **figura 6**.
In questo caso il programma si limita a presentare a video (con la consueta procedura Desktop) i compiti del progetto e a orientare le risorse del computer verso i due servizi, organizzati per visualizzare sulla periferica costituita dai 2 Dot Matrix 8x8 LED Display:
a) due caratteri (o simboli semigrafici) in modo statico;
b) un'informazione scorrevole, Colonna dopo Colonna, da destra verso sinistra.
La scelta è operata nel Main Loop (cioè è affidata alle istruzioni comprese tra le due etichette xxx00) mediante la pressione, rispettivamente, del tasto "F" ("Immagine Fissa") e del tasto "S" ("Immagine Scorrevole"); è ovviamente prevista anche la necessaria via d'uscita: dopo la pressione del tasto "Q" ("Quit") il programma esce dal ciclo principale e, proseguendo via _Quit, viene terminato, restituendo il controllo al sistema operativo.
La **figura 7** analizza il codice delle procedure principali; la prima (Colonna) provvede a gestire completamente l'accensione degli 8 LED di una Colonna di display a matrice di punti, pilotati dalla porta Dati 0378H della parallela e da una delle 16 linee d'uscita di un decoder 74LS154 posto sulla porta 037AH; nella sua semplicità essa si presta benissimo per insegnare le buone tecniche di programmazione, studiate per rendere generica la sua azione, cioè il fatto di affidare sempre la sua funzionalità a dei parametri; nel nostro caso è necessario assicurarsi che Colonna funzioni per ognuna delle possibili 8 Colonne e per qualunque informazione ad essa relativa, per cui è saggio prevedere due parametri: il byte da scrivere sulla Colonna da servire (atteso, in ingresso, in AL) e il numero d'ordine della Colonna stessa (atteso in AH e lasciato inalterato in uscita).
Il primo parametro è immediatamente passato al Registro Dati 0378H/0278H della parallela e da essa agli 8 LED della Colonna corrente; il secondo (un numero binario puro da 0000 a 1111, formattato a 8 bit in AH) esprime la posizione della Colonna corrente nell'ambito di entrambe le matrici, cioè sarà compreso tra

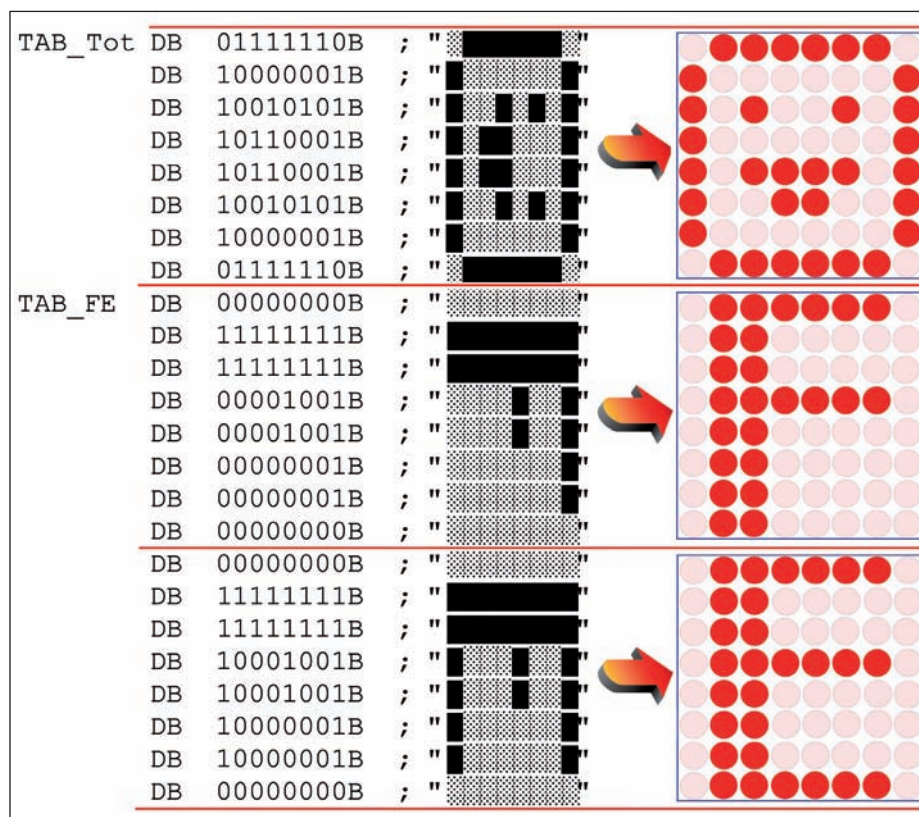



Figura 8: variabile Tab_Tot, una Tabella con i 3x8 byte relativi alle Colonne di 3 caratteri.

AH=00H e AH=07H per quella più significativa (a sinistra) e tra AH=08H e AH=0FH per la rimanente (a destra); questa premessa può bastare nel caso di utilizzo di una porta di microcontrollore, ma non quando deve essere interpretata dal Registro di Controllo 037AH/027AH, dato che 3 dei suoi 4 bit sono inverti internamente (vedi **figura 5**) per cui è necessario provvedere alla ricostruzione dell'informazione da fornire realmente al Registro, cosa facilmente ottenibile con una logica OREX di AH con 00001011B, che "rovescia" il valore dei soli bit3, bit1 e bit0, come richiesto.

Il byte così ottenuto può essere scritto sul Registro di Controllo 037AH e, da esso, passato al decoder 74LS154 che forzerà a massa (tra le sue 16 linee d'uscita) sola quella che corrisponde al codice binario fornito da programma in AH, accendendo di fatto tutti i LED attivi nella Colonna corrente, con l'aiuto del driver Darlington non invertente ULN2981.

Prima di tornare al programma chiamante la procedura Colonna congela le informazioni fornite su entrambe le porte per un tempo sufficiente (un millisecondo) per garantire una loro adeguata visualizzazione, sostenuta dal fenomeno della persistenza ottica.

Per quello che riguarda la seconda procedura principale (matrice), è facile capire che essa si occupa della gestione completa delle 8 Colonne della matrice di sinistra (se chiamata con AH=00H) o di destra (con AH=08H), coinvolgendo 8 volte la prima procedura (Colonna) dopo averle passato il codice binario con la configurazione di LED da accendere, prelevato da una tabella di 8 byte; il puntatore al primo di essi deve essere predisposto in ingresso in SI dalla procedura chiamante.

La **figura 8** mostra la struttura di queste Tabelle, di norma raccolte sotto il nome di una o più variabili, l'etichetta delle quali identifica il primo byte del gruppo, sempre organizzato in batterie di 8, essendo questo il numero delle Colonne da servire in ogni matrice 8x8; la traduzione grafica della sequenza di bit (proposta a destra dei singoli DB, Define Byte) aiuta a capire il loro effetto sui LED, nella Colonna corrispondente, e può servire anche per trovare il giusto metodo per scrivere la Tabella stessa, creando prima l'immagine desiderata e poi "traducendola" in valori binari, 0 per LED spento e 1 per LED acceso. In chiusura analizziamo anche il codice assembly dei due Servizi garantiti da progetto (vedi **figura 9**) attivate dalla pressione dei tasti "F" e "S", nell'ambito del Main Loop. La procedura Display gestisce entrambe le matrici di punti accendendo tutti i 64+64 LED dei due display con le informazioni assunte dagli 8+8 byte raccolti nella tabella puntata in ingresso da SI; nell'esempio suggerito dal Main program di **figura 6**, la Tabella puntata è Tab_FE e il display mostrerà in modo statico il logo della nostra rivista preferita…

Il compito è naturalmente assolto dalla procedura "matrice", chiamata internamente due volte, una per l'accensione dei 64 LED della matrice di punti più significativa (puntata in ingresso con AH=00H) e l'altra per quella a destra (per la quale il puntatore di colonna AH è già pronto al valore AH=08H al termine del servizio precedente). Da notare che, per garantire una necessaria visualizzazione (per "persistenza ottica") della desiderata coppia di caratteri, il processore deve eseguire questa sequenza all'infinito; per questo è necessario appoggiarsi alla Funzione 01H della procedura a livello BIOS nota come INT 16H, che consente di verificare senza fermarsi (JZ Display) la eventuale pressione di tasti sulla tastiera del PC; non appena viene rilevata la pressione di un tasto qualunque la procedura viene terminata (RET) imponendo di fatto il rientro nel Main Loop, per cambiare comando o per terminare anche il nostro progetto. La **figura 9** mostra anche la procedura Scroll, studiata per gestire in modo dinamico l'informazione assunta dalla Tabella puntata in ingresso da SI; per ottenere l'effetto di scorrimento del messaggio da destra verso sinistra è necessario chiamare nx8 volte (con n pari al numero dei caratteri coinvolti) la procedura Display appena descritta, spostando ogni

PER *approfondire...*

- http://www.giobe2000.it/FareElettronica16.htm

```
Display:

        MOV   AH,00

        CALL  Matrice
        CALL  Matrice

        MOV   AH,01H
        INT   16H
        JZ    Display

        RET
```

```
Scroll:

        MOV   DI,SI
        MOV   DX,CX

Scr_00: PUSH  SI

        MOV   AH,00

        CALL  Display

        DEC   CX

        POP   SI
        INC   SI

        CMP   CX,0000H
        JNZ   Scr_01
        MOV   SI,DI
        MOV   CX,DX

Scr_01: MOV   AH,01H
        INT   16H
        JZ    Scr_00

        RET
```

Display

Attiva il servizio per la Matrice del display più significativo (quello a sinistra)

Provvede all'accensione dei 64+64 Led di entrambi i display a Matrice di punti

Provvede a ripetere la scansione completa dei display a Matrice di punti senza fermarsi, fino alla pressione di un tasto

Scroll

Salva il puntatore alla Tabella delle Colonne e il numero di Colonne da visualizzare

Attiva il servizio per la Matrice del display più significativo (quello a sinistra)

Provvede all'accensione dei 64+64 Led di entrambi i display a Matrice di punti

Decrementa il numero di bytes ancora da servire

Aggiorna il puntatore alla Tabella delle Colonne per puntare la locazione successiva al fine di ottenere un effetto spostamento da destra verso sinistra

Se tutte le Colonne sono state servite si reinizializza il puntatore all'inizio della Tabella e il numero di Colonne da visualizzare e si ripete lo scorrimento

Provvede a ripetere la scansione completa dei display a Matrice di punti senza fermarsi, fino alla pressione di un tasto

Figura 9: Dot Matrix 8x8 LED Display - Procedure assembly dei servizi di visualizzazione statica (Display) e scorrevole (Scroll).

Figura 10: nella "finestra" del visualizzatore con due Dot Matrix 8x8 LED Display passa la sequenza delle 24 Colonne previste, reiterata più volte.

volta il puntatore dell'informazione SI di un byte verso il basso. La procedura Scroll può essere migliorata cercando qualche artificio software per variare la velocità di scorrimento (ora in effetti piuttosto rapida), però possiede già un certo grado di generalizzazione per il fatto di affidare la sua funzionalità a due parametri:
a) l'indirizzo iniziale della stringa da visualizzare (cioè della sequenza di byte/Colonna da far scorrere attraverso la "finestra" che rappresenta le 16 Colonne fisiche del display, marcata in arancione nella **figura 10**), atteso, in ingresso, in SI;
b) il numero di byte (Colonne) da coinvolgere nel servizio, atteso in CX e sostanzialmente pari a nx8, sempre con n pari al numero dei caratteri 8x8 che si desidera far circolare. Nell'esempio suggerito dal Main program questa procedura è chiamata a soddisfare le informazioni della tabella identificata dall'etichetta Tab_Tot, in grado di codificare tre caratteri e quindi tale da raccogliere i 24 byte associati alle 24 Colonne necessarie: l'effetto di questa procedura è quello di farne passare il valore attraverso la "finestra". Come si può notare, non appena tutte le Colonne previste sono state servite, la procedura Scroll provvede in proprio a ripuntare lo stesso messaggio iniziale, ricaricando SI con l'indirizzo di Tab_Tot e ricaricando CX con il valore 24; in questo modo la stringa di tre caratteri continuerà a scorrere felice attraverso le Colonne del nostro visualizzatore, almeno fino a quando verrà rilevata la pressione di un tasto qualunque: anch'essa si avvale della Funzione 01H di INT 16H, già descritta per l'altro Servizio. La versione completa del codice Assembly (OutMatrx.ASM, insieme a numerose note esplicative e all'eseguibile, pronto per l'uso, ottenuto dalla sua compilazione) è disponibile puntando il link di approfondimento consigliato. ❏

CODICE MIP 2831584

di MARK CUNDLE, RS COMPONENTS

Le alternative ALLA MEMORIA FLASH

La memoria Flash gode ancora di ottima salute. I dispositivi consumer portatili e intelligenti, come i tablet multimediali e gli smartphone, determinano largamente le tendenze di mercato per le memorie Flash e i semiconduttori in generale, e la quota di mercato dei tablet sembra destinata ad aumentare nei prossimi anni. La previsione di alcuni analisti è che il mercato del dispositivo NAND Flash, il tipo più comune di memoria Flash, avrà un CAGR del 7% fra il 2011 e il 2015.

In termini tecnologici, i produttori vogliono spingersi ancora più là, con maggiori densità di memoria e tecnologie di produzione a 25 nm o meno. I principali produttori di NAND Flash stanno iniziando a realizzare memorie a 64 Gbit in tecnologie di produzione variabili da 20 a 30 nm, utilizzando sempre più spesso strutture e architetture di memoria innovative per soddisfare le esigenze di alta densità.
Un recente documento di SanDisk e Toshiba, presentato nel corso dell'anno alla ISSCC (International Solid State Circuits Conference), ha illustrato in modo dettagliato il dispositivo NAND Flash a 128 Gbit in tecnologia a 19 nm con 3 bit per cella.

POSSIBILITÀ DI SOSTITUZIONE

Mentre le memorie Flash continuano la loro riduzione progressiva nel medio e lungo termine, c'è una costante richiesta di eventuali sostituti a lungo termine, sia per le applicazioni autonome sia per quelle integrate. La competizione include trenta diverse tecnologie di memoria non volatile, che sono vagliate dalle aziende specializzate in semiconduttori, dagli istituti di ricerca e dalle università, con diverse tecnologie che prevedono anche una fornitura in volumi relativamente piccoli. Quattro dei concorrenti principali, che offrono diversi vantaggi rispetto alle memorie Flash, ad esempio tempi di lettura/scrittura cento volte più rapidi e una resistenza per i cicli di scrittura significativamente superiore, sono: la memoria PCM (Phase-Change Memory), la memoria FeRAM o FRAM (Ferroelectric Random Access Memory), la memoria MRAM (Magnetoresistive Random Access Memory) e la RAM resistiva (RRAM).

MEMORIA PCM O PRAM (PHASE-CHANGE MEMORY)

La memoria PCM sfrutta il comportamento del vetro calcogenuro, che è in grado di organizzarsi in quattro stati diversi: cristallino, amorfo e due stati parzialmente cristallini. Il passaggio da uno stato all'altro è indotto dal calore prodotto dal passaggio della corrente elettrica attraverso il vetro. Dato che una memoria PCM può produrre fino a quattro stati, essa raddoppia la capacità di memorizzazione fornendo due bit per cella. Stati diversi hanno resistività elettriche diverse, perciò, ad esempio, lo stato amorfo ad alta resistenza può rappresentare uno '0' binario, mentre lo stato cristallino a bassa resistenza può rappresentare un '1'.
Lo svantaggio principale della memoria PCM è la sensibilità alle alte temperature. Si degrada con l'utilizzo, ma molto più lentamente rispetto a una memoria Flash, che ha una durata nominale di circa 5.000 cicli di scrittura. La memoria PCM può garantire alte prestazioni, specialmente nelle applicazioni di scrittura intensiva perché offre tempi di commutazione rapidi, la possibilità di cambiare i bit senza dover prima cancellare blocchi di memoria e, attualmente, è in grado di raggiungere dimensioni di 40 nm.
Tra Micron Technologies e Samsung è in corso un'accesa competizione per produrre la prima parte di una PCM a 1 Gbit; Samsung finora è riuscita a produrre una PRAM a 512 Mbit compatibile con memoria Flash NOR per i palmari.

MEMORIA FRAM O FERAM (FERROELECTRIC RANDOM ACCESS MEMORY)

La memoria FRAM ha una struttura simile alla memoria DRAM: la DRAM consiste in un trans-

istor di accesso dielettrico e un nodo di storage basato su condensatore, mentre la FRAM utilizza materiale ferroelettrico, come il titanato di piombozirconio (PZT), per formare un condensatore ferroelettrico integrato nel corpo del gate del transistor di accesso.
La struttura cristallina del materiale ferroelettrico permette la formazione di dipoli elettrici semipermanenti, che si allineano in base alla direzione del campo elettrico esterno applicato e mantengono la polarizzazione dopo che il campo è stato rimosso. Questo consente l'accesso casuale a ciascun bit per le operazioni di lettura e scrittura con gli '0' e '1' binari memorizzati, sfruttando le due possibili polarizzazioni disponibili in ogni cella.
La memoria FRAM utilizza meno energia, ha tempi di scrittura più rapidi e consente un numero di cicli di scrittura-cancellazione ben superiore rispetto alla memoria Flash. Gli svantaggi includono basse densità di memoria, limitata capacità di storage e costi elevati.
Fujitsu ha annunciato dispositivi FRAM progettati per rimpiazzare le memorie Flash e SRAM nelle applicazioni industriali, di fabbrica e a bassa potenza, mentre Texas Instruments ha iniziato a offrire microcontrollori basati su FRAM.

MEMORIA MRAM (MAGNETORESISTIVE RANDOM ACCESS MEMORY)

La MRAM utilizza elementi di archiviazione magnetici formati da due piastre ferromagnetiche, entrambe in grado di immagazzinare un campo magnetico, separate da un sottile strato isolante. La struttura più semplice di una cella di memoria è quella di una valvola a spin. Una delle piastre è un magnete permanente a polarità fissa, mentre l'altra ha un campo variabile che si adatta al campo esterno. I bit sono memorizzati rappresentando un '1' con le piastre della stessa polarità e uno '0' con le piastre di polarità opposta. Una griglia di celle di questo tipo forma un dispositivo di memoria. Questa tecnologia promette di sostituire potenzialmente non soltanto la memoria Flash, ma anche la DRAM e la SRAM. La MRAM, tuttavia, è sensibile all'interferenza delle applicazioni di coppia di lunga durata causate da campi magnetici esterni C.C.
Nel 2012, Everspin, spin-off di Freescale Semiconductor, prevede di distribuire un volume di unità MRAM dell'ordine di milioni, ma sta già sviluppando una tecnologia MRAM di seconda generazione, chiamata SST-MRAM (Spin Transfer Torque MRAM). Questa tecnologia sostituisce lo strato isolante con una barriera al tunnelling e utilizza elettroni allineati (polarizzati). Il vantaggio principale è la riduzione della quantità di corrente necessaria per scrivere, cosa che la rende paragonabile al processo di lettura e consente di avere densità più elevate, ma la coerenza di spin deve essere mantenuta e il funzionamento ad alta velocità richiede comunque l'utilizzo di una corrente più elevata. Al momento, con le nuove strutture in fase di sperimentazione, si prevede di utilizzare questa tecnologia a meno di 65 nm.

RAM RESISTIVA (RRAM O RERAM)

La RRAM si basa sulla commutazione elettronica (indotta da corrente o tensione) di un materiale utilizzato come resistenza fra due stati resistivi stabili (basso/alto). Questo viene fatto con l'improvvisa conduzione attraverso isolatori a ossido. La RRAM passa da uno stato resistivo all'altro tramite due operazioni: RESET, che ripristina uno stato di alta resistenza da uno di bassa resistenza, e SET, che fa il contrario.
La RRAM può raggiungere meno di 30 nm e uno studio condotto sulle RRAM basate su ossidi ha persino suggerito che il movimento di ossigeno potrebbe avvenire a 2 nm. Secondo l'Istituto di ricerca IMEC, i dispositivi RRAM con una struttura a elementi sovrapposti potrebbero essere immessi sul mercato a 11 nm, con la Flash 'SONOS' come fase intermedia ai nodi da 17 e 14 nm. La RRAM offre una velocità di commutazione inferiore al nanosecondo e un ridotto consumo energetico e garantisce la stabilità dei dati nonostante le alte temperature e l'usura. La sua robustezza offre nuove opportunità in mercati come quello automobilistico e nelle applicazioni integrate.
Elpida, ad esempio, ha già sviluppato prototipi di RRAM, e ha l'obiettivo di produrne grandi volumi nel 2013, con dispositivi che propongono capacità dell'ordine di gigabit basati su processi costruttivi a 30 nm.

E ANCORA NON SI VEDE LA FINE...

Tuttavia, nessuna di queste quattro tecnologie è ancora riuscita a raggiungere una produzione di massa: sono tutte limitate a mercati di nicchia. La memoria Flash, dal canto suo, sembra in grado di continuare a ridurre le proprie dimensioni ancora per svariate generazioni, perciò pare proprio che passerà ancora un bel po' di tempo prima di poter dire: "Il re è morto, viva...". Il resto lo conoscete.

In questa quarta puntata approfondiremo l'interfacciamento di contatti, parlando di tecniche di debouncing di tipo hardware e firmware. In particolare, coglieremo l'occasione per introdurre la Button Library. Completeremo quindi la discussione avviata nella scorsa lezione sulle funzioni di delay e sul costrutto di ciclo FOR, introdurremo il costrutto CASE e presenteremo, tramite esempi applicativi, le routine della Sound Library.

Come fatto nelle precedenti puntate, proseguiamo lo studio del MikroPascal PRO for PIC, cercando di bilanciare adeguatamente la trattazione teorica del linguaggio con esempi pratici di programmazione e con riferimenti circuitali utili a una comprensione organica della materia anche da parte di coloro che hanno poca esperienza in merito. Questo approccio dovrebbe aiutare il lettore a non incorrere in una acquisizione dei concetti approssimativa che possa portare a un'applicazione pratica sommaria, se non addirittura poco corretta di quanto appreso.
Per questo cercheremo anche di bilanciare la presentazione di aspetti propri del linguaggio, come la sintassi, con concetti e aspetti di carattere più direttamente applicativi, spesso legati all'impiego di procedure e funzioni di libreria. Prendendo spunto da quanto esposto nella precedente lezione, approfondiremo l'interfacciamento di contatti parlando di tecniche di debouncing sia di tipo hardware sia di tipo firmware. In questo ambito introdurremo al lettore la Button Library. Completeremo quindi la discussione avviata nella scorsa lezione sulle funzioni di delay e sul costrutto di ciclo FOR e illustreremo l'impiego del costrutto CASE e delle routine della Sound Library.

IL DEBOUNCING

Il debouncing è uno dei primi accorgimenti che è necessario adottare quando si acquisiscono in ingresso al controllore stati logici derivanti dalla commutazione di contatti, come quelli di interruttori, pulsanti o relè elettromeccanici. Il problema è concettualmente banale ma, se non tenuto in debita considerazione, può costituire causa di malfunzionamenti delle applicazioni firmware tutt'altro che trascurabili. Infatti, la commutazione meccanica o elettromeccanica di un contatto può generare una transizione dell'ingresso del controllore da uno stato all'altro, passando elettricamente per un certo numero di transizioni spurie. Il tutto diventa tanto più evidente quanto meno rapida e netta è la chiusura o l'apertura del contatto. Il fenomeno è per questo ancora più evidente in presenza di interruttori e pulsanti manovrati manualmente.
Per meglio comprendere il fenomeno, si faccia riferimento a un pulsante o un interruttore inserito all'interno di un circuito a pull-up come quello di **figura 1**. La transizione elettrica, dallo stato alto a

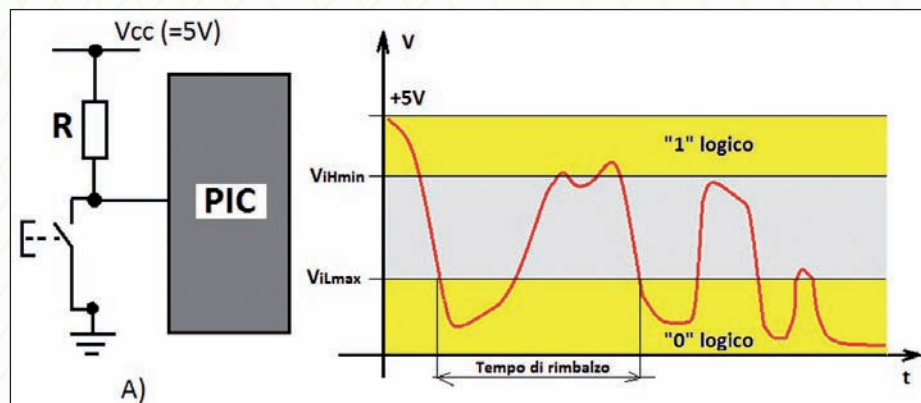



Figura 1: transizioni elettriche spurie sull'ingresso del PIC durante la commutazione di un contatto. Quando il segnale è superiore a V_{IHmin} il segnale viene interpretato come livello logico "1", quando invece è inferiore a V_{ILmax} è interpretato come livello logico "0" (nella fascia compresa tra questi valori, l'interpretazione è incerta). Il rimbalzo elettrico del contatto produce pertanto un rimbalzo dello stato logico rilevato.

di ANTONIO GIANNICO

quello basso durante la chiusura del contatto e viceversa, e da quello basso a quello alto durante l'apertura, non è ideale ma passa generalmente per una serie di rimbalzi elettrici che dipendono essenzialmente dalla tipologia del contatto, dal fatto che si sta interrompendo eventualmente una corrente pregressa che non può azzerarsi istantaneamente e dalla modalità con cui lo stesso contatto viene manovrato e quindi commutato. Poiché un pin di un controllore rileva come "0" logico lo stato elettrico se questo è inferiore a un certo valore V_{ILMAX} e "1" logico se questo è superiore a un certo valore V_{IH-MIN}, l'andamento spurio del segnale **(figura 1)** viene interpretato come una sequenza di transizioni da stato logico "0" a stato logico "1" e viceversa, prima che si raggiunga una situazione stabile. Il tempo che intercorre tra l'istante in cui il contatto viene interpretato come commutato per la prima volta e quello in cui tale interpretazione diventa definitiva è detto "tempo di rimbalzo". Il significato fisico di questo parametro è quello per cui è necessario, dopo la prima transizione, attendere questo tempo affinché lo stato letto sul pin del controllore sia effettivamente stabile. L'analisi del fenomeno tramite un semplice oscilloscopio mostra frequentemente, al posto di una commutazione netta e continua, il passaggio da uno stato all'altro attraverso un treno di impulsi più stretti e ripidi. Al fenomeno contribuiscono, tra le altre cose, gli effetti propri di rimbalzo meccanico della lamina di contatto e il degradamento, per invecchiamento o per l'accumulo di sporcizia o ossidazione, della capacità di condurre corrente elettrica da parte del contatto.

La fisica del fenomeno elettrico del rimbalzo e quindi il tempo di rimbalzo, come già detto, dipendono essenzialmente dalla tipologia, dal modello del contatto e dalla modalità con cui esso viene manovrato. Fissati questi parametri tuttavia, il tempo di debouncing non può essere considerato deterministico poiché ripetendo l'evento, per esempio di chiusura manuale di un pulsante, potrebbero manifestarsi tempi di rimbalzo differenti da una ripetizione all'altra. Va inoltre considerato, tra i parametri che determinano il tempo di rimbalzo, il regime di funzionamento del circuito in cui il contatto è inserito, il suo invecchiamento e quindi, in generale, anche le condizioni ambientali. Il tempo di rimbalzo assume in ogni caso, nei circuiti elettronici, tipicamente valori compresi tra pochi millisecondi e alcune decine di millisecondi.

La cosa basilare da comprendere è che un rimbalzo elettrico può falsare completamente la corretta acquisizione di uno stato logico da parte di un controllore e che questo può a sua volta falsare l'evoluzione del flusso firmware. Il circuito a microcontrollore, in altre parole, assume un funzionamento del tutto inaffidabile, non essendo il controllore messo nelle condizioni di distinguere correttamente le commutazioni reali da quelle spurie. Il rimbalzo potrebbe far evolvere lo stato del processo firmware rapidamente dallo stato 0 allo stato 2, quando invece il comando realmente impartito prevedeva la sola transizione dallo stato 0 allo stato 1. Per avere un'idea di quanto una situazione del genere rischi di essere frequente nelle applicazioni, si pensi a quanta elettronica è interfacciata verso all'utente attraverso pulsanti. È necessario quindi "filtrare" in qualche modo gli eventi di rimbalzo elettrico rendendoli innocui. A tale scopo si adottano le cosiddette tecniche di debouncing. Alcune di queste sono di tipo prettamente hardware, altre sono invece di natura firmware. Le seconde possono essere implementate sia direttamente dal programmatore sia facendo uso di funzioni di libreria. Di seguito ne illustreremo alcune e ne analizzeremo brevemente vantaggi e svantaggi, al fine di dare al lettore gli strumenti teorici e pratici per applicarli efficacemente.

DEBOUNCING HARDWARE

Dal momento che la logica digitale in elettronica è sempre più spesso di tipo programmabile, il problema del debouncing viene gestito prevalentemente per via firmware. In ogni caso è anche possibile gestire, in una certa misura, il problema con accorgimenti di tipo hardware. La scelta del tipo di circuito di debouncing da utilizzare è legata, in questo caso, principalmente al tipo di contatto (pulsante singolo o doppio, relè elettromeccanico) e alla reale necessità di filtrare i rimbalzi stessi in funzione del modo in cui il segnale che transita viene utilizzato dal circuito. A seconda dei casi si possono utilizzare sia circuiti analogici impieganti resistori e condensatori sia circuiti digitali a porte logiche o anche soluzioni ibride.

Come già detto, il rimbalzo elettrico si presenta spesso sotto forma di impulsi brevi e ripidi, motivo per cui il suo spettro è concentrato alle alte frequenze. Questo suggerisce di contrastare il problema mediante l'impiego di un filtro passa-basso e quindi in sostanza di un circuito RC che, per sua natura, tende a eliminare le componenti in alta frequenza. Tradurre in implementazione circuitale questo accorgimento significa semplicemente inserire, in parallelo al contatto, un condensatore di capacità adeguata **(figura 2B)**. Il valore della resistenza di limitazione e della capacità del condensatore devono essere tali da imprimere al circuito RC una costante

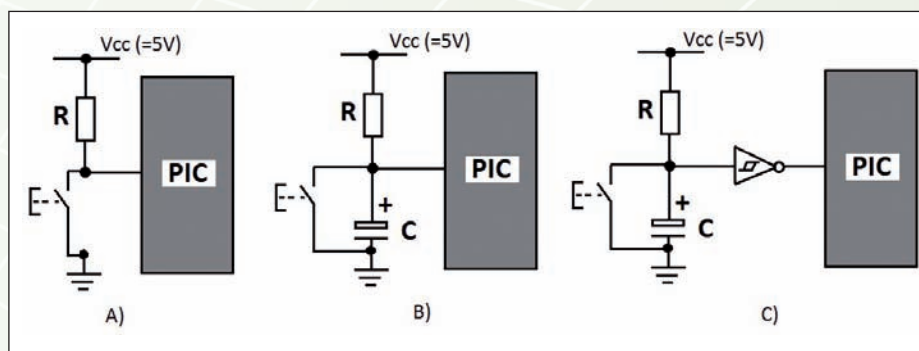



Figura 2: A) Circuito soggetto a rimbalzo elettrico; B) circuito con debouncing RC; C) circuito con debouncing RC e Trigger di Shmitt.

di tempo T=RC dell'ordine del tempo di debouncing (tipicamente alcune decine di millisecondi). Detto in termini più semplicistici, se il contatto risulta aperto la capacità è carica; chiudendo il contatto il potenziale su di esso segue la scarica del condensatore, contrastando effetti di rimbalzo; allo stesso modo, chiudendo il contatto, giunti a un certo valore di carica, un rimbalzo di ritorno verso il livello logico zero non può avvenire poiché risulta impedito dall'effetto di tenuta del condensatore che imprime una certa inerzia al rimbalzo stesso contrastandolo. A tale scopo è anche possibile porre una piccola resistenza in serie al pulsante stesso nel circuito di **figura 2B**.
Il debouncing così ottenuto può essere reso ancora più efficace inserendo tra il circuito RC e il pin del controllore un circuito comparatore a Trigger di Shmitt, in grado di introdurre un effetto di squadratura finale del segnale grazie all'introduzione di un'isteresi che filtra di fatto le residue oscillazioni. Il circuito assume allora l'aspetto di **figura 2C**. I vantaggi di un tale approccio, utilizzato dai circuitisti soprattutto in passato, quando le logiche firmware erano meno diffuse, risiedono soprattutto nel ridotto costo implementativo legato all'elevata semplicità circuitale.
Per contro, la tecnica non consente una soluzione totale del problema in presenza di commutazioni particolarmente rumorose, specie in quei casi in cui l'aumento della costante di tempo RC a valori troppo elevati, nel tentativo di rendere sicura l'interpretazione, porta a introdurre ritardi non accettabili nell'esecuzione del processo.
Una differente tecnica di debouncing hardware è quella che prevede l'interfacciamento del controllore al contatto attraverso un latch SR **(figura 3)**. Allo scopo possono essere utilizzate porte NAND oppure porte NOR. L'antirimbalzo si ottiene grazie al fatto che un latch SR mantiene stabile lo stato dell'uscita indipendentemente dal numero di impulsi ricevuti sul corrispondente ingresso, almeno fino a quando tali impulsi interessano un solo ingresso. Circuitalmente, l'unico inconveniente sta nel fatto che, per invertire il segnale in uscita, l'impulso di commutazione va applicato su due diversi ingressi (il Set e il Reset appunto), come mostrato in **figura 3**. Ne consegue un'immunità al disturbo maggiore rispetto a quanto è possibile ottenere con i circuiti analogici RC, a scapito, tuttavia, di una complessità circuitale leggermente superiore. Il suo impiego dipende da diverse considerazioni alcune delle quali di opportunità (si pensi al caso in cui nella la realizzazione dello schema elettrico del circuito a microcontrollore, almeno due porte NAND o NOR di un integrato restino libere e quindi utilizzabili allo scopo, senza l'aggiunta di ulteriore componentistica attiva). Facendo riferimento a un latch SR a porte NAND, quello che accade può essere dedotto dall'analisi del circuito di **figura 3** e della tabella di verità dello stesso circuito **(tabella 1)**.
La linea sulla quale si vuole rilevare la presenza di un "1" è quella sull'ingresso S negato. Posto che R negato si trovi connesso a GND attraverso il contatto (S negato è quindi a livello alto), lo stato dell'uscita è di reset, cioè Q=0. In queste condizioni, una commutazione del contatto comporta su R negato, durante il distacco, il passaggio, anche ripetuto, per una condizione di memoria per cui l'uscita resta a livello "0". Appena S negato "tocca" per la prima volta lo stato zero (il contatto si sta chiudendo su S negato appunto), l'uscita scatta a livello alto "1". Eventuali instabilità su S durante la chiusura vengono nuovamente neutralizzate per l'effetto di memoria del latch.
L'efficacia del latch SR come antirimbalzo sta quindi tutta nella sua intrinseca capacità di mantenere uno stato di memoria in presenza di specifiche commutazioni. Senza dilungarci oltre segnaliamo che, sebbene di uso meno frequente, anche i circuiti monostabili possono essere utilizzati come circuiti di debouncing. Questi presentano oltretutto il vantaggio di non necessitare di doppio segnale (Set e Reset).

Figura 3: circuito antirimbalzo a latch RS. Si tenga in considerazione la tabella di verità del latch (tabella 1) per una corretta interpretazione del suo funzionamento e dell'effetto di debouncing che se ne ottiene.

TABELLA 1

$\bar{S}$	$\bar{R}$	Q
0	0	N.V. (non valida)
0	1	1
1	0	0
1	1	Q (Memoria)

Tabella 1: tabella di verità di un Latch SR con porte NAND. Si noti come l'effetto di debouncing sia reso possibile dalla condizione di memoria.

DEBOUNCING FIRMWARE

Compreso il modo in cui il fenomeno si manifesta, appare ovvio pensare, come tecnica da adottare per contrastare il rimbalzo elettrico, a quella che si basa su tre semplici operazioni:
1) intercettare la prima transizione dello stato logico;
2) attendere alcune decine di millisecondi che le transizioni spurie si esauriscano;
3) acquisire lo stato logico che a questo punto può essere considerato stabile.
L'operazione al punto 2 è quella fondamentale per la riuscita della strategia in quanto è quella che consente di filtrare temporalmente le transizioni spurie in maniera che esse vengano ignorate dal punto di vista del flusso firmware del microcontrollore. Ovviamente, è necessario fissare in maniera appropriata il tempo di ritardo con cui eseguire il filtraggio temporale dei rimbalzi. Il tempo di attesa deve infatti essere più lungo di quello di

```
while true do
  begin
  if Button(PORTB, 0, 1, 1) then oldstate := 255;
  if oldstate and Button(PORTB, 0, 1, 0) then
      begin
       PORTD := not(PORTD);
       oldstate := 0;
      end;
   end;
```

Figura 4: esempio di impiego della funzione di libreria Button. Quando RB0 transita da livello logico "1" a livello logico "0", la porta D viene invertita nello stato logico dei suoi bit. RB0 deve essere stato preventivamente configurato come ingresso mentre la porta D deve essere stata preventivamente configurata come uscita.

rimbalzo ma più breve del tempo per il quale lo stato che segue la transizione permane disponibile sul pin di ingresso del controllore. Una routine che agisce secondo la logica appena esposta può essere scritta abbastanza agevolmente dal programmatore. Nel caso si impieghi MikroPascal PRO for PIC è anche possibile utilizzare la funzione di libreria Button, appositamente pensata per interfacciare contatti e di cui parleremo nel successivo paragrafo. Ovviamente, un blocco di codice che esegue il controllo dello stato logico su un determinato pin di ingresso, secondo la logica appena esposta, comporta una limitazione. Infatti, se il controllo dell'ingresso è inserito in un codice più ampio che esegue anche altre operazioni, l'attesa necessaria per filtrare i rimbalzi potrebbe inficiare la corretta esecuzione con la corretta tempistica delle altre operazioni. In questo caso potrebbe essere opportuno intercettare la transizione sfruttando la capacità del PIC di generare delle interruzioni (interrupt) in occasione di un cambiamento di stato che si manifesta sui pin della porta B. Nel caso del PIC 16F877, per esempio, è possibile sfruttare in tal senso il pin RB0/INT oppure uno dei pin RB4, RB5, RB6 ed RB7 (interrupt on change). Ci limitiamo, per il momento, a quanto detto; l'impiego di interrupt verrà trattato in maniera più dettagliata in successive puntate del corso. Una tecnica altrettanto efficace e sostanzialmente equivalente di debouncing è quella di effettuare letture successive in rapida sequenza dello stato del segnale senza interrompere con attese il flusso di programma e decidere che la commutazione è avvenuta quando i valori letti si stabilizzano.
Appare evidente come il principale svantaggio applicativo delle tecniche di debouncing hardware, rispetto a quella firmware, stia nel fatto di dover calibrare adeguatamente la rete RC e il trigger di Shmitt e nel dover prevedere componentistica ulteriore on-board, con conseguente aumento dell'area occupata dal circuito.
Per contro, una soluzione firmware è certamente più flessibile poiché può essere facilmente adeguata al comportamento dello specifico contatto e del suo tempo di debouncing senza introdurre complicazioni circuitali.

```
for i:= 10 do
    begin
       temp := a[i];
       a[i] := b[i];
       b[i] := temp;
       i := i + 1;
    end;
```

Figura 5: esempio di ciclo iterativo FOR applicato a un blocco di codice che scambia ordinatamente il contenuto di un vettore a di 10 elementi con quello di un vettore b di pari dimensione.

BUTTON LIBRARY

La Button Library rende disponibile una routine per rilevare la pressione di un pulsante e prevenire, via firmware, gli effetti di rimbalzo che possono verificarsi in tali occasioni. La libreria è composta dalla sola funzione Button, il cui prototipo è il seguente:

function Button (var port : byte; pin, time, active_state : byte) : byte;

La funzione restituisce rispettivamente:
- il valore 255 se il pin è nello stato attivo per un determinato periodo;
- 0 negli altri casi.

Essa consente in tal modo di filtrare l'effetto di instabilità, cioè di rimbalzo del contatto, consentendo un efficace antibouncing. Per un corretto impiego della funzione è necessario specificare rispettivamente:
- la porta cui appartiene il pin su cui eseguire l'acquisizione con debouncing (parametro port);
- il pin di interesse all'interno di detta porta (parametro pin che può assumere valori nel range 0..7);
- il parametro time che rappresenta il periodo di debouncing espresso in millisecondi;
- il parametro active_state che può assumere valore 0 oppure 1 a seconda che il contatto o pulsante che determina fisicamente la transizione dello stato logico sia attivo sullo stato logico 0 oppure sullo stato logico 1.

Ovviamente è necessario, per il suo corretto impiego, configurare il pin di interesse come ingresso. Se, per esempio, si vuole monitorare il pin RB0, sul quale si suppone che sia connesso un pulsante, e si intende leggerne lo stato, supponendo che si tratti di un pulsante normalmente aperto il cui stato di chiusura determini una transizione da livello 1 a livello 0, si dovrà scrivere un blocco di codice come quello riportato in **figura 4**.
Il blocco di codice presentato in questo esempio inverte lo stato della porta D al verificarsi della transizione. Ovviamente, affinché il tutto funzioni correttamente è anche necessario che la porta D sia stata preventivamente configurata come uscita. È importante osservare che il tempo inserito nell'istruzione (parametro time) genera un'attesa a tutti gli effetti anche per le istruzioni che seguono, la cui esecuzione, di conseguenza, risulta ritardata. Per questo motivo la routine va impiegata con attenzione. Si sottolinea inoltre come il parametro time è anche esso di tipo byte, per cui può assumere valori compresi tra 0 e 255 (millisecondi). Non è di utilità pensare a valori superiori dal momento che il rimbalzo elettrico tipicamente non raggiunge mai valori dell'ordine delle centinaia di millisecondi.

LE ROUTINE DI BUILT-IN

Tra le diverse categorie di routine di libreria che MikroPascal PRO for PIC mette a disposizione vi sono le cosiddette routine di built-in. Esempi di routine di built-in sono Delay_us e Delay_ms, utilizzate negli esempi della scorsa puntata. Si tratta, in altri termini, di routine implementate a

A)
```
if espressione1 then
  if espressione2 then istruzione1
    else istruzione2;
```

B)
```
if espressione1 then
   begin
     if espressione2 then istruzione1
       else istruzione2
   end;
```

C)
```
if espressione1 then
   begin
    if espressione2 then istruzione1
   end
   else istruzione2;
```

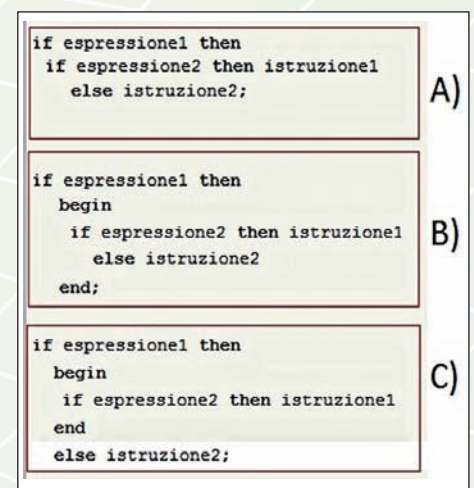



Figura 6: la porzione di codice riportata in A) viene intesa dal compilatore come esplicitato in B). Completamente differente è invece, dal punto di vista algoritmico e quindi esecutivo, il significato della porzione di codice C).

livello di codice compilato come blocchi "inline", nel senso che il relativo blocco compilato viene collocato direttamente in corrispondenza del punto di chiamata della routine. Ciò nasconde un piccolo dettaglio che è opportuno sottolineare.
Da un punto di vista di firmware compilato aggiungere una serie di Delay_ms o Delay_us in un codice di programma non è come aggiungere una chiamata ma è piuttosto come aggiungere altrettante volte la stessa porzione di codice compilato. Questo aspetto può essere indagato facilmente, scrivendo un semplice programma che contenga tra le parole riservate begin ed end sole istruzioni delay_us, per esempio. Se ne aumentiamo li numero vediamo crescere proporzionalmente la quantità di memoria di programma occupata, come si deduce osservando, nell'area di notifica Messages, i valori riportati alle voci "Used ROOM (program words)" e "free ROOM (program words)" in seguito a compilazione del codice. D'altro canto, come si può facilmente verificare, l'impiego delle routine delay_ms o delay_us non necessita di alcuna spunta all'interno della Library Manager, contrariamente a quanto necessario quando si impiega una funzione di libreria. Ciò accade proprio perché si tratta di routine di built-in.

Routine di delay Vdelay_Advanced_ms e Delay_Cyc

Nella scorsa lezione abbiamo preso confidenza con le routine di Built-in Delay_ms, Delay_us e Vdelay_ms. Quest'ultima può accettare, come argomento, tanto una variabile word quanto una costante che rientri nel range di variazione di una word (corrispondente a 16 bit). Mentre Vdelay_ms può quindi essere utilizzata al posto di Delay_ms, non è vera la cosa reciproca. Questo è del resto facilmente riscontrabile nel modo in cui, all'interno dell'help in linea vengono definiti i prototipi delle due routine:

- procedure **Delay_ms**(time_in_ms: const dword);
- procedure **Vdelay_ms**(time_in_ms : word)

Completiamo l'analisi delle routine di delay parlando delle routine Vdelay_Advanced_ms e Delay_Cyc.
Vdelay_Advanced_ms. *È una procedura il cui prototipo è il seguente:*
procedure ***VDelay_Advanced_ms****(time_ms, Current_Fosc_kHz: word);*
Non restituisce alcun valore e consente di generare un ritardo firmware di durata, espressa in millisecondi, (ms) pari al valore della variabile time_ms. Il valore del delay generato non è accurato come quello ottenuto con Delay_ms. Per esempio, il seguente blocco di codice:
pause := 1000;
fosc := 10000;
VDelay_Advanced_ms(pause, fosc);
consente di ottenere un ritardo firmware di circa 1000 ms = 1 secondo.
Delay_Cyc. *È una procedura il cui prototipo è il seguente:*
procedure ***Delay_Cyc****(Cycles_div_by_10 : byte);*
Essa non restituisce alcun valore e consente di generare un ritardo basato sulla frequenza di clock. Il ritardo dura un tempo pari a dieci cicli di MCU. Una limitazione nel suo impiego sta nel fatto che il valore che il parametro Cycles_div_by_10 può assumere deve appartenere al range 2..257.
Per esempio, la seguente riga di codice:
Delay_Cyc(10);
consente di ottenere 10 volte un ritardo pari a dieci cicli di MCU cioè, nel complesso, un ritardo pari a 100 cicli di MCU.

Approfondimento sulle routine di delay: __Lib_Delays.mpas

È piuttosto interessante analizzare il listato del file **__Lib_Delays.mpas**, che è possibile trovare all'interno della directory Mikroelektronika\mikroPascal PRO for PIC\Uses\P16 per i controllori della serie 16 e in Mikroelektronika\mikroPascal PRO for PIC\Uses\P18 per i controllori della serie 18. L'analisi del listato è interessante per due principali motivi:
1) consente di comprendere più in profondità come siano strutturate e implementate le procedure di delay Delay_Cyc, VDelay_ms e VDelay_Advanced_ms e come esse contengano istruzioni asm come "nop" (no operation) per implementare il ritardo;
2) consente di osservare come sia semplice implementare una procedura che renda possibile generare un ritardo di qualunque durata utilizzando i due soli "mattoni fondamentali" Delay_us e Delay_ms.
Questa analisi aiuta inoltre a comprendere perché, sebbene Delay_Cyc, VDelay_ms e VDelay_Advanced_ms siano a volte riportate nell'help in linea tra le routine di built-in, sono in realtà più simili a vere e proprie funzioni di libreria.

CLASSIFICAZIONE DELLE ISTRUZIONI

Nelle precedenti lezioni abbiamo accennato alla struttura generale di un codice MikroPascal e abbiamo cominciato a utilizzare concretamente le prime istruzioni. Il concetto ci è ormai abbastanza familiare. Cerchiamo di approfondire la questione. Le istruzioni di un linguaggio di programmazione come il MikroPascal definiscono, e quindi codificano, mediante costrutti ad alto livello, azioni ben precise all'interno di un algoritmo e consentono di formare, nel complesso, l'algoritmo stesso. Ogni istruzione si basa su una specifica sintassi. A prescindere dalla sintassi della specifica istruzione, il compilatore ha bisogno di riconoscere le istruzioni al fine di poterle codificare in linguaggio macchina attraverso la cosiddetta compilazione. A tale scopo, esso necessita di distinguere, prima di tutto, l'inizio e la fine di ciascuna istruzione. In MikroPascal, l'elemento terminatore che consente questa operazione è il punto e virgola (;), pertanto ciascuna istruzione si distingue dalla successiva grazie all'interposizione di questo segno di punteggiatura.
Da un punto di vista esecutivo, terminata un'istruzione, il controllore esegue la successiva, a meno che la sequenzialità del flusso di programma non sia intanto stata modificata da un'istruzione di salto.
Da un punto di vista della classificazione delle istruzioni possiamo più precisamente distinguere:

- istruzioni di assegnazione (Assignment Statements);

```
case selettore of
    valore_1 : istruzione_1;
     ...
    valore_n : istruzione_n;
    [else istruzione_default]
end;
```

Figura 7: sintassi dell'istruzione di selezione "Case...of...". Nel caso in cui venga utilizzato il ramo else, l'ultima istruzione che lo precede deve terminare senza il punto e virgola.

- istruzioni condizionali (Conditional Statements);
- istruzioni di iterazione (Iteration Statements o Loop);
- istruzioni di salto (Jump Statements) tra cui è possibile annoverare le chiamate a procedure (procedure call);
- istruzioni assembler (asm Statement).

Esistono poi i cosiddetti blocchi di istruzioni (Compound Statements) che non rappresentano, tuttavia, un ulteriore tipo di istruzione.
Ciascuna istruzione, o blocco di istruzioni, può a sua volta essere inserita in altre istruzioni strutturate dello stesso tipo tra cui salti, loop e diramazioni condizionate. Di seguito analizziamo singolarmente le tipologie di istruzioni appena elencate. Questa analisi è interessante poiché aiuta a comprendere come l'insieme di istruzioni di un linguaggio di alto livello sia più elaborato di quello di un linguaggio di basso livello, specie per quanto attiene il controllo del flusso.

Istruzioni di assegnazione (Assignment Statements)

Un valore può essere assegnato solo ad una variabile in quanto contenitore di valori cui è riservata fisicamente, dal compilatore, una ben precisa locazione di memoria. La forma più generale che un'istruzione di assegnazione può assumere in MikroPascal è la seguente:

variabile := espressione;

È importante osservare che non si è scritto "variabile :=valore;" che rappresenta solo un sottocaso della precedente ma "variabile := espressione" poiché con il termine espressione si intende, più in generale, un'espressione che consente di calcolare un valore, a partire da determinati suoi argomenti. Ovviamente, la variabile destinata a contenere il valore deve appartenere a un tipo opportunamente dichiarato, e lo stesso dicasi per le variabili contenute all'interno dell'espressione.
È importante osservare che l'operatore di assegnazione ":=" è diverso dall'operatore "=" che invece è un operatore di relazione impiegato in istruzioni che consentono di verificare l'uguaglianza, per esempio, tra il valore contenuto in una variabile e una costante.
È opportuno notare che l'operatore è anche differente da quello di "=" utilizzato per la dichiarazione delle costanti (cambia il contesto; avremo modo di comprendere meglio questo aspetto al momento opportuno) e da quello di ":" utilizzato per la dichiarazione delle variabili.
Avere ben chiari e distinti questi concetti ci mette fin da subito nelle condizioni di non stilare con fare incerto codici dal comportamento esecutivo alquanto imprevedibile, nel caso in cui non sia il compilatore stesso a segnalare errori o warning in fase di compilazione.

Istruzioni condizionate (Conditional Statement)

Le istruzioni condizionate (o istruzioni di selezione) consentono di direzionare il flusso di codice in una direzione piuttosto che un'altra, in base al verificarsi di una condizione piuttosto che un'altra. Nella scorsa lezione abbiamo preso confidenza con l'istruzione IF-THEN-ELSE; in realtà MikroPascal, come molti altri linguaggi strutturati, prevede due tipi di istruzioni condizionate:

- istruzione if..then..else;
- istruzione case..of..;

Approfondiamone i tratti distintivi al fine di poterne effettuare un corretto impiego.

Istruzione IF

Questo tipo di istruzione condizionata è stata già trattata nella scorsa lezione. Ci limitiamo per questo a ricordarne la sintassi:
*if espressione **then** istruzione_1 [**else** istruzione_2];*
Con le due parentesi quadre stiamo indicando che la porzione di codice contenuta potrebbe anche non essere presente. In questo caso (e solo in questo) istruzione_1 termina con il punto e virgola. Al posto delle due istruzioni potremmo anche avere dei blocchi di codice. In questo caso l'esecuzione dell'intero blocco risulta condizionata al verificarsi della condizione. È inoltre utile sottolineare come "espressione", dal punto di vista esecutivo del codice, altro non è che un valore booleano risultato dell'elaborazione stessa dell'espressione. Inoltre è importante ricordare che la parola riservata else non deve mai essere preceduta da punto e virgola (;) che indica invece la fine di una istruzione, pena l'inevitabile errore di compilazione che MikroPascal notifica all'interno dell'area Messages.

Istruzioni IF annidate

L'impiego di istruzioni annidate richiede una certa attenzione in quanto un cattivo uso può determinare un comportamento scorretto del programma senza che vi sia formalmente alcun errore dal punto di vista della sintassi così come analizzata dal compilatore. Per esempio, l'espressione di **figura 6A** viene intesa dal compilatore come riportato in **figura 6B**, che è qualcosa di completamente differente da quanto riportato in **figura 6C**. L'errore per questo non può essere "scovato" dal compilatore in fase di compilazione, essendo tutti e tre i pezzi di codice perfettamente corretti da un punto di vista sintattico, ma deve essere prevenuto direttamente dal programmatore facendo attenzione al significato, da un punto di vista algoritmico, di quanto si sta scrivendo.

Blocchi di codice (Compound Statements)

Un blocco di codice è una sequenza di istruzioni comprese tra le parole chiave begin ed end:
begin
sequenza di istruzioni
end;
Vale la solita regola per cui ciascuna istruzione all'interno del blocco si conclude con un punto e virgola.
Da un punto di vista del flusso di programma, un blocco di istruzioni può a tutti gli effetti essere considerato come un'unica macroistruzione. Il blocco può allora essere condizionato alla pari di una singola istruzione mediante il costrutto IF, oppure essere inserito in istruzioni di ciclo. Per esempio, un ciclo FOR potrebbe rappresentare l'iterazione su un intero blocco di codice, come mostrato nell'esempio di **figura 5**. Blocchi possono inoltre essere nidificati l'uno dentro l'altro.
Quanto mostrato per il ciclo FOR, in merito ai blocchi di codice, può esser detto in relazione alle istruzioni di controllo "IF.. THEN..ELSE", " WHILE...DO" e "REPEAT...UNTIL" (le ultime due saranno trattate in lezioni successive). Si noti come lo stralcio di codice mostrato come esempio in **figura 5** altro non sia che un ciclo che consente di scambiare gli elementi di un vettore a di dieci elementi con i corrispondenti elementi del vettore b composto da altrettanti elementi.
L'operazione è possibile solo impiegando

```
case operator of
  '*' : result := n1 * n2;
  '/' : result := n1 / n2;
  '+' : result := n1 + n2;
  '-' : result := n1 - n2
  else result := 0;
end;
```
A)

```
case reg of
  0: opmode := 0;
  1,2,3,4: opmode := 1;
  5,6,7: opmode := 2;
end;
```
B)

```
case byte_variable of
  byte_var1: opmode := 0;
  byte_var2: opmode := 1;
  byte_var3: opmode := 2;
end;
```
C)

Figura 8: esempi di impiego dell'istruzione "Case...of...". È possibile riunire sotto uno stesso ramo più valori separando gli stessi con virgole come nell'esempio B). In mikroPascal PRO for PIC, i valori che compaiono all'interno della lista valori del costrutto case possono anche essere delle variabili (figura 8C).

una variabile di passaggio cioè temporanea (temp) che contenga momentaneamente il valore dell'elemento i-esimo del vettore destinazione prima che lo stesso venga sovrascritto dal corrispondente elemento dell'altro vettore. Quest'ultima è dimensionata per contenere un solo elemento del vettore. Ovviamente si potrebbe utilizzare come variabile temporanea anche un intero vettore. La differenza sta nella quantità di memoria dati allocata, cioè riservata dal compilatore durante la compilazione. Avremo occasione in altre puntate di approfondire l'impiego di dati strutturati, in particolare dei vettori, anche detti array.

Istruzione Case

Anche il costrutto Case è un'istruzione di diramazione del flusso che presuppone l'analisi di una serie di condizioni. Si tratta per questo di una diramazione multipla equivalente a tutti gli effetti a un selettore, a ciascuna posizione delle quali corrisponde un valore, quindi una condizione e pertanto un ramo del flusso. La sintassi dell'istruzione è mostrata, nella forma più generale, in **figura 7**. Chiaramente, come al solito, al posto di istruzione_n potremmo anche avere un intero blocco di istruzioni. Il selettore è una espressione che può assumere, nei casi più comuni, valori interi oppure corrispondere a un carattere. Il ramo else è opzionale e può essere omesso nel caso in cui non sia prevista una condizione di default. Nel caso in cui questo ramo venga utilizzato non deve essere posto il punto e virgola prima dell'else (esattamente come accade con l'else dell'istruzione IF-THEN-ELSE). L'espressione selettore viene prima valutata e, solo a questo punto, il suo valore può essere confrontato con quelli corrispondenti a ciascun ramo del flusso. Nel caso in cui l'espressione dia come risultato uno dei valori dei rami del costrutto, la corrispondente istruzione (o blocco di istruzioni) viene eseguita e l'intero case ha termine. Se vi sono diversi rami la cui condizione è verificata, solo la prima di esse viene effettivamente eseguita mentre, nel caso in cui alcun valore venga a corrispondere con quello assunto dall'espressione, viene eseguita l'istruzione di default che compare in fondo alla lista dopo la parola riservata else (ammesso che questa opzione venga contemplata e non omessa nel costrutto). In **figura 8** riportiamo alcuni esempi. È anche possibile riunire sotto uno stesso ramo più valori, separando gli stessi mediante virgole (**figura 8B**). Sottolineiamo come il fatto di utilizzare, come valori di selezione, interi o caratteri è solo una consuetudine frequente legata anche al fatto che variabili di questo tipo occupano due soli byte nel primo caso e un solo byte nel secondo. In realtà, in MikroPascal PRO for PIC, i valori di selezione e il risultato dell'elaborazione dell'espressione possono essere valori integer, byte, char, short, word, dword o longint. Inoltre, tanto il selettore quanto i valori di selezione possono anche essere delle variabili (**figura 8C**) oppure espressioni contenenti variabili. Tutto ciò rende il costrutto estremamente flessibile. Per il momento non ci dilunghiamo oltre. Avremo modo di ampliare e completare il discorso nella prossima lezione quando, alla luce di quanto introdotto fino a ora, dedicheremo un paragrafo della lezione ai tipi di dati semplici, cioè non strutturati, al fine di averne ben chiare le differenze.

Istruzioni di iterazione (Iteration Statements o Loop)

Le istruzioni appartenenti a questa categoria consentono di eseguire l'iterazione di una o più istruzioni creando, di fatto, un loop controllato. I costrutti che consentono di operare in tal senso sono in MikroPascal i seguenti:

- For;
- While…do…;
- Repeat…until…

All'interno delle istruzioni iterative è possibile utilizzare le istruzioni break e continue, che consentono di aggiungere un ulteriore controllo a flusso di codice e più precisamente al loop. Break determina la conclusione del ciclo e quindi l'uscita immediata dal loop mentre continue determina l'esecuzione immediata della successiva iterazione. Per il momento, di questi ci limitiamo a illustrare il solo ciclo FOR.

```
var somma, i,j,k:integer;
begin
TRISB:=0;
      .
      .
      .
  { Main program }
  somma := 0;
  j:=1;
  k:=4;
for i := (j) to (k+2) do
begin
  somma := somma + a[i] + b[i];
  k:=k-1;
  if (somma=12) then portB.0:=1
  else portB.0:=0;
  delay_ms(100);
  end
end.
```

Figura 9: valore_finale e valore_iniziale sono valori dello stesso tipo di contatore; vengono ricalcolati ogni volta che il ciclo ricomincia. Può trattarsi di valori interi definiti oppure di variabili o espressioni funzioni di variabili intere i cui valori possono essere modificati all'esterno o all'interno interno del blocco di codice eseguito nel loop.

Ciclo For

Il ciclo FOR implementa un'iterazione che richiede che sia specificato il numero di iterazioni. La sintassi dell'istruzione è la seguente:

for contatore := valore_iniziale to valore_finale do lista_ istruzioni

oppure

for contatore := valore_iniziale downto valore_finale do lista_istruzioni

Contatore è una variabile intera che incrementa o decrementa il suo valore a ogni ciclo a seconda che venga utilizzata la prima sintassi (to) o la seconda (down-

Figura 10: listato del codice di esempio Esempio1-Sound. Si noti in particolare: 1. la modularità del codice ottenuta attraverso l'impiego di procedure di libreria o definite dal programmatore; 2. l'antibouncing ottenuto attraverso l'impiego della routine della Button Library; 3. le modalità di impiego delle procedure della Sound Library.

to). Prima della prima iterazione, il contatore viene impostato al valore iniziale, nel corso delle iterazioni invece viene incrementato o decrementato fino a raggiungere il valore finale, evento che determina l'uscita dal ciclo. Si sottolinea che per variabile intera si intende, in questo caso, una variabile i cui valori possono essere ordinati in maniera crescente o decrescente. Può trattarsi pertanto di integer, byte, char, short, word, dword e longint ma non di real. Nel caso in cui si dichiari la variabile contatore come real, in fase di compilazione, il contatore notifica infatti il seguente errore "304 Syntax error: Expected integral type but real found". Valore_finale e valore_iniziale, che sono valori dello stesso tipo di contatore, vengono ricalcolati ogni volta che il ciclo ricomincia. In altre parole, può trattarsi di valori interi definiti oppure di una variabile oppure di una espressione di valore intero (funzione di variabili di valore intero) collocata all'esterno o all'interno del blocco di codice eseguito nel loop **(figura 9)**.
Valore_finale e valore_iniziale possono inoltre essere espressioni calcolabili al momento della compilazione (per esempio la somma di un valore e una costante definita in testa al listato di codice) oppure calcolabili solo a runtime all'esterno del ciclo prima del suo inizio.
Se la lista di istruzioni è composta da più di una istruzione allora il blocco è racchiuso tra le parole chiave begin ed end. Di seguito un esempio nel quale viene calcolato il prodotto scalare tra due vettori a e b di lunghezza 10.

```
s := 0;
for i := 0 to 9 do
s := s + a[i] * b[i];
```

Il ciclo iterativo ha termine quando il contatore raggiunge o supera valore_finale. È importante osservare che un modo, a volte più leggibile, di creare cicli è quello di utilizzare il costrutto While che può anche assumere la forma di ciclo infinito "While TRUE… do…", oppure il costrutto Repeat di cui parleremo in lezioni successive.

SOUND LIBRARY

MikroPascal PRO for PIC dispone di una libreria (Sound Library) che può essere utilizzata dal programmatore per generare suoni all'interno delle applicazioni. A parte la generazione del segnale, va detto che affinché il suono possa essere ascoltato è necessario disporre di uno speaker, per esempio un piezo-speaker connesso opportunamente al controllore, come mostrato in **figura 11**.
La libreria è composta essenzialmente di due funzioni:

- Sound_Init;
- Sound_Play.

Le analizziamo brevemente al fine di chiarirne le corrette modalità di impiego.

Sound_Init

Si tratta di una procedura il cui prototipo risulta il seguente:
*procedure **Sound_Init**(var snd_port : byte; snd_pin : byte);*
Essa consente di configurare il corretto pin del controllore per la generazione del suono e non restituisce alcun valore. Per una corretta configurazione è necessario definire i seguenti parametri:

- snd_port: porta di uscita su cui generare il suono;
- snd_pin: pin della suddetta porta su cui generare il suono.

Per esempio, con la seguente istruzione:
Sound_Init(PORTC, 3);
si configura per la generazione del suono la porta RC3.

Sound_Play

Anche in questo caso si tratta di una procedura il cui prototipo è il seguente:
*procedure **Sound_Play**(freq_in_hz, duration_ms : word);*
Essa consente di generare un'onda quadra su di uno specifico pin e non restituisce alcun valore. Per un corretto impiego è necessario specificare i seguenti parametri:

- freq_in_hz: frequenza del segnale (onda quadra) espresso in Hertz (Hz);
- duration_ms: durata del segnale alla frequenza data espressa in millisecondi (ms).

La massima frequenza, espressa in Hertz, che è possibile generare dalla funzione è:
Freq_max = Fosc/(80*3)
mentre la minima è data da
Freq_min = Fosc/(80*255)
La frequenza realmente generata può differire da quella impostata attraverso il prametro freq_in_hz a causa di approssimazioni a valori interi che derivano dal processo di generazione della forma d'onda stessa.

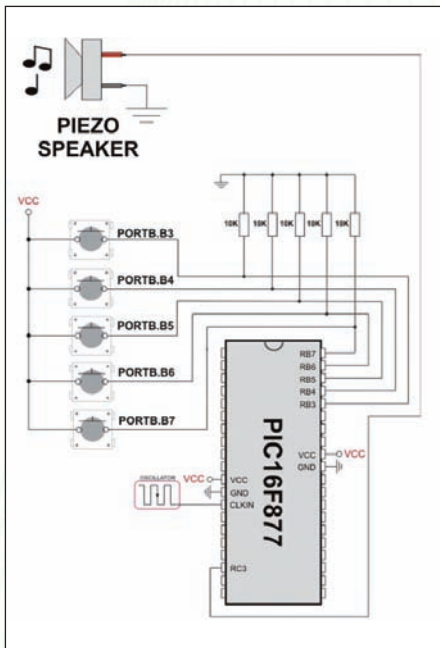



Figura 11-Schema elettrico cui fa riferimento il firmware dell'Esempio 1-Sound il cui listato è riportato in figura 10. Non si è riportato in maniera esplicita il connettore di programmazione del controllore con i relativi jumper di selezione, la parte del circuito oscillatore esterna al controllore e il pulsante di reset. Per tali dettagli si faccia riferimento al circuito di figura 13 che, pur essendo relativo all'esempio successivo, non differisce dal presente per quanto riguarda questi dettagli [4].

La routine può essere adeguatamente impiegata solo se si effettua preventivamente la necessaria configurazione attraverso la procedura Sound_Init. Per esempio scrivendo:

Sound_Play(1000, 100);

dopo l'inizializzazione (routine Sound_Init), si genera, a runtime, un'onda quadra di frequenza 1000 Hz = 1 kHz e durata temporale 100 millisecondi = 0,1 secondi.
I codici di esempio mostrati nel successivo paragrafo mostrano, tra le altre cose, l'impiego della routine Sound_Play(...). Essi sono in parte tratti dall'Help in linea di MikroPacal Pro for PIC versione 5.61. La loro analisi ci consente, da un lato comprendere meglio l'impiego delle procedure di libreria Sound_Init() e Sound_Play() e dall'altra di cominciare a prendere confidenza con la strutturazione modulare di un programma ottenuta mediante chiamata di procedure e/o funzioni di libreria o scritte dallo stesso programmatore.

ESEMPI APPLICATIVI

Concludiamo questa quarta puntata del coso proponendo tre semplici esempi applicativi che consentono di analizzare e verificare, da un punto di vista pratico, quanto fin qui esposto. Come al solito, il lettore potrà modificarli a piacimento al fine di acquisire maggiore famigliarità con i concetti.

Esempio 1-Sound

Il codice di programma **(figura 10)** mostra un esempio di impiego delle procedure della Sound Library. Si nota, in particolare, il corpo principale del programma (ciclo while TRUE) che contiene una serie di istruzioni IF le quali non fanno altro che intercettare il pulsante premuto al fine di dare avvio alla generazione di una ben precisa nota o melodia che presenti una ben precisa durata.
L'antibouncing è effettuato utilizzando la procedura Button della Button Library. Il ciclo while che segue ciascuna generazione di tono o di melodia serve a consentire all'utente il rilascio del pulsante. Ciascun tono e ciascuna melodia richiamata all'interno del ciclo principale di programma è preventivamente definita come procedura. Ciò conferisce al codice una struttura ordinata e modulare. Importanti risultano le inizializzazioni che riguardano l'impiego della procedura di libreria Sound_play e la definizione degli ingressi del controllore. Per lo schema elettrico di riferimento dell'applicazione si rimanda alla **figura 11**.

Esempio 2-SquareGen

Il codice **(figura 12)** mostra come sia semplice, attraverso l'impiego della Sound Library, implementare un generatore di onda quadra con frequenza dell'onda impostabile. La frequenza viene selezionata attraverso i pin RB0, RB1, RB3 sui quali insistono tre microswitch con la possibilità di codificare, pertanto, 8 combinazioni e quindi 8 differenti frequenze dell'onda quadra. I tre bit vengono estratti dalla porta B grazie a una operazione di AND bit a bit (bitwise) con la maschera 00000111.
Il codice può essere modificato facilmente e ampliato a tutte le possibili combinazioni degli otto bit componenti la porta di ingresso B. Si tratta di un modo diverso ma anche più efficiente di scrivere

```
program SquareGen;
var set_T:byte;
begin
  TRISB := 0xFF;
  TRISA := 0x00;
  Sound_Init(PORTA, 0);
  while TRUE do
    begin
     set_T:=(PORTB)and(%00000111);
     case set_T of
      0: Sound_Play(100, 1000);
      1: Sound_Play(500, 1000);
      2: Sound_Play(1000, 1000);
      3: Sound_Play(2000, 1000);
      4: Sound_Play(5000, 1000);
      5: Sound_Play(5000, 1000);
      6: Sound_Play(7500, 1000);
      7: Sound_Play(10000, 1000);
    end;
    end;
end.
```

Figura 12: esempio di generatore d'onda quadra implementato attraverso l'impiego della Sound Library. La frequenza viene selezionata attraverso i pin RB0, RB1, RB2 sui quali insistono tre microswitch. Possono in questo modo essere codificate 8 differenti frequenze dell'onda quadra.

una applicazione già presentata nella scorsa lezione. Per lo schema elettrico dell'applicazione si faccia riferimento a **figura 13**.

ESEMPIO 3-SQUAREGEN2

Questo esempio **(figura 14)** mostra come un codice del tipo mostrato in Esempio 1 possa essere reso ancora più compatto. Qualunque sia il valore intero impostato sulla porta B, sul pin RA0 è generata un'onda quadra di frequenza pari al valore intero contenuto in PortB moltiplicato per cento.
L'impiego del piezo-speaker è legato soprattutto al fatto che un componente del genere assorbe una corrente di pochi milliampere. Esso può pertanto essere collegato direttamente al controllore senza particolari accorgimenti. Normalmente, si tratta di dispositivi di piccole dimensioni dall'impedenza che va da alcune centinaia di ohm alle frequenze più alte (una decina di chilohertz circa), ad alcuni chiloohm alle frequenze più basse (alcune centinaia di hertz). Diversamente, l'impiego di un piccolo altoparlante o di uno speaker di altra natura necessiterebbe per lo meno di essere collegato ai pin del controllore attraverso un transistor che riceva dal controllore il solo segnale e che provveda a fornire potenza allo speaker per la generazione del segnale acustico.

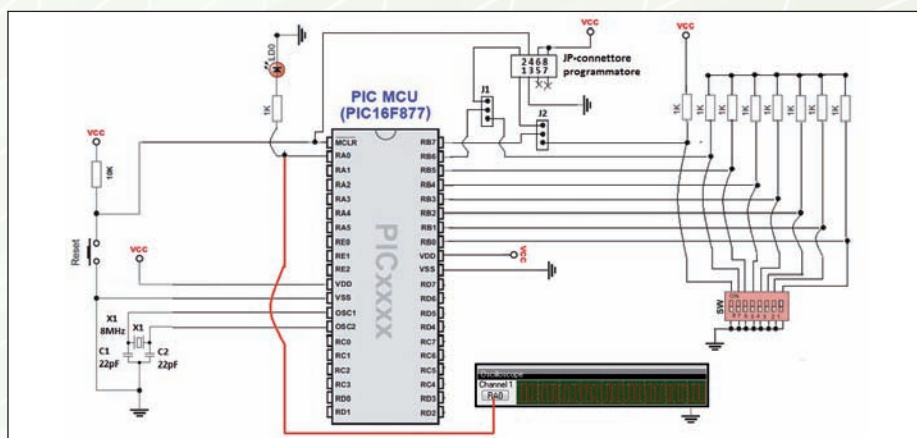



Figura 13: schema elettrico di riferimento per le applicazioni firmware Esempio 2-SquareGen (figura 12) ed Esempio 3-SquareGen2 (figura 14).

```
program SquareGen2;
{ Dichiarazione variabili }
begin
  TRISB := 0xFF;
  TRISA := 0x00;
  Sound_Init(PORTA, 0);
  while TRUE do
    begin
      if (PORTB<>0) then Sound_Play(PORTB*100, 1000);
    end;
end.
```

Figura 14: esempio 3-SquareGen2. Il codice consente di generare un'onda quadra la cui frequenza espressa in hertz è pari al valore impostato sulla porta B moltiplicato per 100.

Figura 15: si noti come l'impiego di una routine di libreria necessiti di essere inserita nel progetto attraverso la Library Manager (segno di spunta). In caso contrario, pur essendo il listato di programma corretto da un punto di vista sintattico, il compilatore genera un errore in fase di compilazione definendo la routine come non dichiarata (Identifier ... was not declared).

Come è possibile osservare dall'analisi dei listati di esempio, le procedure possono essere considerate a tutti gli effetti dei sottoprogrammi. Il loro impiego facilita l'organizzazione modulare del codice e quindi un approccio strutturato e di tipo top-down, utile anche in fase di modifica o manutenzione del firmware da parte dello stesso programmatore o di programmatori terzi. Volendo fare un utile esercizio pratico si provi a stilare il codice dei suddetti esempi senza fare uso delle procedure composte dal programmatore ma utilizzando le sole procedure di libreria, come la procedura Sound_play, direttamente inserite nel blocco principale di programma. Vi accorgerete facilmente di come il codice diventi meno leggibile.
Ancora due brevi osservazioni:
1) la prima riguarda la posizione in cui vengono collocate le funzioni definite dal programmatore; esse devono sempre precedere il blocco principale di codice;
2) la seconda riguarda l'impiego delle routine di libreria. È infatti sempre necessario spuntare nella Library Manager la libreria di cui si fa uso. Per esempio, se nell'impiegare la Sound Library questa non viene spuntata all'interno della Library Manager, il compilatore non saprà di doverla utilizzare e la compilazione non andrà a buon fine **(figura 15)**. Questa spunta all'interno della Library Manager è quindi a tutti gli effetti una sorta di direttiva per il compilatore.

CONCLUSIONI

In questa quarta lezione del corso MikroPascal PRO for PIC abbiamo imparato che qualora si voglia acquisire in ingresso a un controllore lo stato logico di contatti o pulsanti, che per loro natura possono essere soggetti a rimbalzi, è necessario adottare alcuni accorgimenti impiegando tecniche di debouncing di natura hardware e/o firmware. In particolare, chi impiega MikroPascal PRO for PIC può utilizzare le funzioni della Button Library integrata nell'IDE MikroPascal.
Abbiamo anche proseguito nell'analisi delle differenti categorie di istruzioni (istruzioni di assegnazione, istruzioni condizionali, istruzioni di iterazione, chiamate a procedure). L'analisi della struttura dei cicli FOR e del costrutto CASE ci ha consentito di dotarci degli strumenti necessari a scrivere in forma più efficace alcuni codici di esempio che traggono spunto anche da precedenti puntate.
Nel corso dell'esposizione abbiamo avuto a che fare con funzioni e procedure di libreria, oltre che con procedure scritte dal programmatore. Abbiamo così compreso come il loro impiego consenta un approccio modulare alla stesura del codice con tutti i vantaggi che ne derivano in termini di leggibilità.

NELLE PROSSIME PUNTATE

Nel prosieguo del corso avremo modo di approfondire l'argomento della programmazione modulare e di illustrare le differenze che esistono tra procedure e funzioni, di introdurre nuove funzioni di librerie e proporre nuovi esempi applicativi utili alla comprensione dei concetti. Per essere più efficaci, non trascureremo di trattare l'interfacciamento fisico al controllore di dispositivi esterni di I/O e di realizzare applicazioni realmente utili da un punto di vista pratico. ❏

CODICE MIP 2831578

PER *approfondire...*

- [1] PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers (www.mikrochip.com).
- [2] Sito di Microchip (www.mikrochip.com).
- [3] MikroPascal PRO for PIC v.5.61 demo e documentazione relativa (www.mikroe.com).
- [4] Help in linea MikroPascal Pro for PIC v. 5.61.

Lo shop dei prodotti Inware Edizioni...

Nuova piattaforma!

PRIMO PIANO

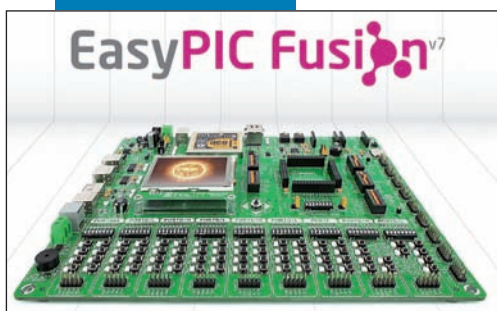
Genius Card

Diventa un membro della community dell'elettronica, acquistando oggi la tua GENIUS membership card. Avrai diritto a ricevere:
1) 11 nuovi numeri di Fare Elettronica sia in edizione cartacea sia in PDF (in coda all'eventuale abbonamento esistente)
2) 11 nuovi numeri della rivista digitale Firmware (in coda all'eventuale abbonamento esistente)
3) Tutto l'archivio delle riviste Firmware pubblicate mensilmente a partire da febbraio 2010
4) Accesso al club della rivista Firmware con centinaia di articoli in PDF
5) Accesso al club della rivista Fare Elettronica (centinaia di articoli in PDF)
6) Due e-book a scelta (Smart card, CPLD, display, PIC, AmpOP, alimentatori, linguaggio C)
7) Due raccolte delle annate di Fare Elettronica (dal 2003 al 2011) e Firmware (dal 2006 al 2011) a scelta.

CODICE: GENIUS-CARD - PREZZO: € 69.50

Scheda di sviluppo easyPIC7 Fusion

EasyPIC Fusion v7 è la nuova scheda di sviluppo di MikroElektronika che unisce il supporto per tre diverse architetture: dsPIC33, PIC24 e PIC32 in una singola scheda. Contiene molti moduli on-board, tra cui moduli multimediali, Ethernet, USB, CAN e altri. Inoltre, on-board il potente programmatore mikroProg e il debugger In-Circuit che supportano 65 microcontroller. La scheda è provvista di un potente programmatore on-board mikroProg e di un debugger In-Circuit che supportano oltre 65 microcontrollori. Il mikroProg è dotato di una programmazione veloce e avanzata e un ricco set di istruzioni di debug. Il debugger è supportato da tutti i compilatori di MikroElektronika. La scheda è dotata di display TFT 320x240 con pannello touch, codec audio stereo, dispositivi USB e host USB, navigation switch e slot per schede microSD sono un insieme di periferiche perfette per una piattaforma di sviluppo multimediale. Il supporto mikroBUS consente di utilizzare decine di schede accessorie "Click" con estrema semplicità senza alcuna regolazione hardware. Aggiungere nuove funzionalità al dispositivo non è mai stato così facile.

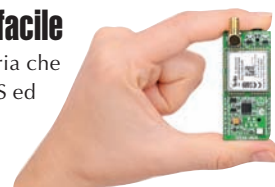
CODICE: EASYPIC7 FUSION - PREZZO: € 192.39

ELEKTOR MEMBERSHIP

NOVITÀ

Sottoscrivi la tua membership ad Elektor e rimani sempre aggiornato nel campo dell'elettronica. In qualità di "GREEN member" riceverai:
• 8 edizioni digitali di Elektor in lingua italiana;
• 2 edizioni speciali digitali "Jumbo Edition" con il doppio delle pagine a gennaio/febbraio e luglio/ agosto
• Uno sconto del 10% su tutti gli articoli disponibili sul sito Elektor
• Accesso ad Elektor.LABS
• Accesso ad Elektor.MAGAZINE
• La ricezione degli Elektor.POST nella tua casella email (con oltre 25 progetti extra ogni anno)
• La tua GREEN card personalizzata

CODICE: MEMBERSHIP ELEKTOR
PREZZO: € 50.00

Usa il GSM/GPRS in modo facile

GSM Click è una scheda accessoria che utilizza il nuovo socket mikroBUS ed è una soluzione perfetta per aggiungere la comunicazione GSM/GPRS al proprio dispositivo. È dotato del modulo GSM/GPRS Telit GL865-QUAD che supporta frequenze Quadri Band GSM/GPRS 850/900/1800/1900 MHz. La scheda contiene il traslatore di tensione bidirezionale a 6 bit TXB0106 e socket SIM Card. La scheda GSM Click comunica con il microcontrollore target tramite sette linee mikroBUS (RX, TX, INT, PWM, CS, RST e AN). GSM Click può utilizzare come tensione di alimentazione 3,3 V o 5V. Un diodo LED indica la presenza di alimentazione.

CODICE: CLICK-GSM - PREZZO: € 47.19

NOVITÀ

WiFi click

WiFi Plus Click è una scheda accessoria che utilizza il nuovo socket mikroBUS. Si tratta di una soluzione compatta per aggiungere la comunicazione WiFi al vostro dispositivo. È dotato di MRF24WB0MA - 2,4 GHz, modulo conforme IEEE std. 802.11, come pure il controller MCW1001 con on-board lo stack TCP/IP e connection manager 802.11. La scheda WiFi Plus Click comunica con scheda di destinazione tramite l'interfaccia UART. La scheda è progettata solo per utilizzare alimentazione a 3,3 V. Esiste, inoltre, un'opzione per ottenere un basso consumo energetico.

CODICE: WIFI-SPI - PREZZO: € 42,35

OFFERTA

Arduino Due

Arduino Due, la nuova arrivata nella famiglia Arduino, è la prima scheda basata su un processore a 32 bit ARM, l'Atmel SAM3X8E ARM Cortex-M3, che migliora tutte le funzionalità standard di Arduino e aggiunge altre nuove caratteristiche.
Possiede 54 pin di I/O digitali (16 dei quali possono essere utilizzate come uscite PWM, con risoluzione selezionabile), 12 Pin di Input Analogici 12 bit di risoluzione, 4 UART (porte seriali hardware), e due uscite DAC (digital to analog converter), un cristallo oscillatore a 84 MHz , due connessioni USB, un jack di alimentazione, un header ICSP, un header JTAG e un pulsante di reset. La tensione massima che i pin I/O sono in grado di fornire o tollerare è di 3.3V, tensioni più elevate, come 5V al pin di input potrebbe danneggiare la scheda.
Arduino Due dispone di due connettori USB, il connettore micro-usb B è quello nativo in grado di agire come host USB, ovvero è possibile collegare periferiche esterne compatibili con USB, come mouse, tastiere, smartphone, mentre l'altra porta USB con il connettore di tipo A è per scopi di debugging.

CODICE: A000062 - PREZZO: € 47,19

(10% DI SCONTO INSERENDO IL COUPON MV873PJ288XMC)

OFFERTA

La nuova scheda EasyPIC7

EasyPIC v7 è la settima generazione della famosa scheda di sviluppo PIC. È il prodotto di conoscenze accumulate nell'arco di 10 anni e rappresenta lo stato dell'arte nella progettazione, funzionalità e qualità. È amata dai neofiti e hobbisti così come dai professionisti e utilizzata nell'insegnamento e sviluppo da migliaia di persone. La scheda viene fornita con un PIC18F45K22.

CODICE: EASYPIC7 - PREZZO: € 175,45

(10% DI SCONTO INSERENDO IL COUPON MV873PJ288XMC)

Programmazione BASIC per PIC

Il libro permette all'utente di programmare i microcontrollori PIC utilizzando l'ambiente di sviluppo Mikrobasic, in modo da ottenere il massimo impiegando il BASIC come linguaggio ad alto livello. Passo dopo passo vengono esplorati tutti i segreti e i trucchi della programmazione con un linguaggio chiaro, semplice ma allo stesso tempo tecnico.
I costrutti del linguaggio e le molteplici tecniche di programmazione vengono illustrati attraverso la realizzazione di progetti pratici completi di schema elettrico e codice sorgente.
Tra gli argomenti trattati troverete la tecnica PWM, la conversione AD, l'uso dei motori passo-passo, la gestione dei suoni, l'uso delle EEPROM oltre alla guida all'uso del compilatore e di tutti i suoi strumenti integrati.

CODICE: FE-27 - PREZZO: € 24.90

(10% DI SCONTO INSERENDO IL COUPON MV873PJ288XMC)

Arduino Starter Kit

Questo nuovo kit vi introdurrà nel mondo di Arduino in maniera molto pratica ed è perfetto per imparare e sperimentare le conoscenze dell'elettronica. Imparerete il facile utilizzo di Arduino mediante la costruzione di diversi progetti creativi. Il kit comprende una selezione dei componenti più comuni e utili e un libro con le istruzioni per 15 progetti pratici. Iniziando dalle basi dell'elettronica , fino a progetti più complessi, questo kit vi aiuterà a controllare il mondo fisico con sensore e attuatori.

CODICE: K000007 - PREZZO: € 96.68

Lo shop dei prodotti Inware Edizioni... ma non solo!

Batteria Li-Polymer 3,7 V 2000mAh

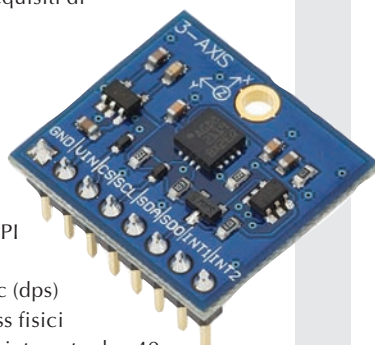
La batteria Li-Polymer è la migliore soluzione per l'aggiunta di alimentazione propria e stabile per i dispositivi che richiedono la mobilità e una fonte di energia a lunga durata. Dispone di connettore standard compatibile con le schede mikromedia. La batteria fornisce 3,7 V e ha una capacità di 2000 mAh.

Applicazioni
Se avete bisogno di una alimentazione a batteria, questa è una buona soluzione e molto compatta. Questo batteria ricaricabile a 3,7 V 2000 mAh è in grado di offrire molto di più.
Caratteristiche principali:
- Funzionalità: 3,7 V 2000 mAh
- Design compatto: 63 x 44 x 6 mm
- Funzionamento stabile e grande curva di performance

Vantaggi principali:
- Grande capacità e lunga durata
- Soluzione economica per un'alimentazione stabile per il prototipo o dispositivo finale.

CODICE: MKE-BATT-LI-POL - PREZZO: € 15061

Modulo giroscopio a tre assi

Il modulo giroscopio L3G4200D è un sensore di velocità angolare a 3 assi a bassa potenza con, inoltre, i dati di temperatura.
I dati della misura della velocità angolare e i dati della temperatura sono accessibili dall'interfaccia digitale selezionabile (I2C o SPI). Il modulo è dotato di un piccolo package e ha un facile accesso all'interfaccia SIP con un foro di montaggio per una veloce connettività con i vostri progetti.
È progettato per l'uso con una grande varietà di microcontrollori con requisiti di tensione differenti.
Caratteristiche:
Sensore di velocità angolare a 3 assi (oscillazione, rollio e beccheggio)
Supporta la comunicazione I2C e SPI
Tre scale selezionabili: 250/500/2000 gradi/sec (dps)
Alta immunità agli stress fisici
Sensore di temperatura integrato da –40 a +185 °F (da –40 a + 85 °C)
Embedded Power-down e sleep-mode
Valori dati output: 16 bit
Dati output Temperatura: 8 bit

CODICE: 27911 - PREZZO: € 34.97

Ethernet card per easyPIC7 Fusion

PIC32MX795F512L con supporto ETHERNET per MCU Card standard a 104 pin per EasyFusion 7 con MCU packaging a 100 pin TQFP e distanza 100 mil.

CODICE: MCU-CARD-PIC32MX795F512L-ETH - PREZZO: € 32.31

MCU card dsPIC33FJ256GP710A

MCU Card dsPIC33FJ256GP710A standard a 104 pin per EasyFusion 7 con MCU packaging a 100 pin TQFP e distanza 100 mil.

CODICE: MCU-CARD-DSPIC33FJ256GP710A PREZZO: € 28.07

SPECIALE PER I LETTORI:
FAI L'ORDINE ONLINE SU WWW.ELETTROSHOP.COM/IESHOP E INSERISCI IL CODICE COUPON 59FC1A09CB39FB12 LA SPEDIZIONE SARÀ GRATIS!

prossimamente su Fare Elettronica

ARDUINO DUE

Ne ha fatta di strada il Team che ha dato il via al progetto Arduino. Nel 2005 nascono i primi prototipi con la scheda diventata ormai famosa in tutto il mondo. Un'idea tutta italiana, nata a Ivrea, già famosa per la sede dell'Olivetti. Dopo vari aggiornamenti, nell'ottobre del 2012 ha visto la luce la nuova versione della scheda, la Arduino DUE, basata sulla CPU Atmel ARM Cortex-M3 SAM3X8E e, la prima della famiglia Arduino, basata su un microcontrollore a 32 bit con core ARM. Vediamo di che cosa si tratta.

JREBOARD PER PIC

Perché non costruire una Jreboard quasi universale che permetta in 5-10 minuti di attivare un circuito (modulare) di test SENZA SALDATORE? Ecco come fare.

PREPARAZIONE DELLA SD PER IL RASPBERRY PI

Un utilissimo articolo di pronto intervento sulla preparazione della SD Card contenente il sistema operativo del Raspberry Pi.

CONTI CORRENTI POSTALI - Ricevuta di Accredito — BancoPosta

€ sul C/C n. 70107552 di Euro

TD 451

importo in lettere

INTESTATO A: INWARE EDIZIONI SRL

CAUSALE

ESEGUITO DA:

VIA - PIAZZA

CAP

LOCALITÀ

BOLLO DELL'UFF. POSTALE codice bancoposta

IMPORTANTE: NON SCRIVERE NELLA ZONA SOTTOSTANTE

importo in euro — numero conto — td

AUT. DB/SISB/E 20017 DEL 11.04.2006

70107552 < 451>

CONTI CORRENTI POSTALI - Ricevuta di Versamento — BancoPosta

€ sul C/C n. 70107552 di Euro

importo in lettere

INTESTATO A: INWARE EDIZIONI SRL

ESEGUITO DA:

VIA - PIAZZA

CAP

LOCALITÀ

BOLLO DELL'UFF. POSTALE

AVVERTENZE
Il Bollettino deve essere compilato in ogni sua parte (con inchiostro nero o blu) e non deve recare abrasioni, correzioni o cancellature.
La causale è obbligatoria per i versamenti a favore delle Pubbliche Amministrazioni.
Le informazioni richieste vanno riportate in modo identico in ciascuna delle parti di cui si compone il bollettino.

CAUSALE

Indagine sui lettori

Aiutaci a conoscerti meglio!

Con il tuo aiuto riusciremo ad offrirti una rivista sempre più in linea con le tue aspettative.

Compila online il questionario all'indirizzo www.farelettronica.com/survey

Per ringraziarti per il tuo tempo e la tua cortesia, ti invieremo gratuitamente un bellissimo eBook del valore di 14,52 euro!

AVVERTENZE

Il Bollettino deve essere compilato in ogni sua parte (con inchiostro nero o blu) e non deve recare abrasioni, correzioni o cancellature.

La causale è obbligatoria per i versamenti a favore delle Pubbliche Amministrazioni.

Le informazioni richieste vanno riportate in modo identico in ciascuna delle parti di cui si compone il bollettino.

PRODOTTI

Se vuoi ordinare "offline" ovvero senza usare Internet, usa il seguente modulo d'ordine **PRODOTTI**

COD. PRODOTTO	DESCRIZIONE	PREZZO UNITARIO	Q.tà	Totale

SUB TOTALE	
SPESE DI SPEDIZIONE	
TOTALE	

Nota. Se le righe non sono sufficienti, utilizzare più copie del modulo.
Per maggiori dettagli sulle spese di spedizione e i metodi di pagamento vedi retro cartolina.

METODI DI PAGAMENTO

☐ **BONIFICO BANCARIO** ☐ **BOLLETTINO POSTALE** ☐ **CONTRASSEGNO**

☐ MasterCard ☐ VISA ☐ postepay

TITOLARE. . . . .
NUMERO . . . .
DATA DI SCADENZA. . . . . CODICE DI SICUREZZA . . . .

INVIARE IL MODULO D'ORDINE A: Inware Edizioni srl, Via Giotto, 7 - 20032 Cormano (MI) - oppure via FAX allo 02 66508225
(Inviando questo modulo si accettano i termini e le condizioni riportate a tergo)

Nome . . . . Cognome . . . .
Indirizzo . . . . n° . . . .
Cap . . . . Città . . . . Prov . . . .
Tel . . . . Fax . . . . Email . . . .
Ragione Sociale . . . .
Codice fiscale . . . . Partita IVA . . . .

DATA . . . . FIRMA . . . . ☐ Fattura

FE 330

ABBONAMENTO

Sì, voglio diventare membro o rinnovare la membership

☐ MAKER MEMBERSHIP: 11 nuovi numeri di Fare Elettronica + Bonus pack "MAKER" (vedi retro) € 55,00
☐ SMART MEMBERSHIP: 11 nuovi numeri di Firmware + Bonus pack "SMART" (vedi retro) € 29,50
☐ GENIUS MEMBERSHIP: 11 nuovi numeri sia di Firmware che di Fare Elettronica + Bonus pack "GENIUS" (vedi retro) € 29,50

METODI DI PAGAMENTO (per maggiori dettagli vedi retro cartolina)

☐ **BONIFICO BANCARIO** ☐ **BOLLETTINO POSTALE** ☐ **ALLEGO ASSEGNO** (Intestato a Inware Edizioni)

☐ MasterCard ☐ VISA ☐ postepay

PayPal fare il pagamento su paypal@inwaredizioni.it

TITOLARE. . . . .
NUMERO . . . .
DATA DI SCADENZA. . . . . CODICE DI SICUREZZA . . . .

INVIARE IL MODULO D'ORDINE A: Inware Edizioni srl, Via Giotto, 7 - 20032 Cormano (MI) - oppure via FAX allo 02 66508225
(Inviando questo modulo si accettano i termini e le condizioni riportate a tergo)

Nome . . . . Cognome . . . .
Indirizzo . . . . n° . . . .
Cap . . . . Città . . . . Prov . . . .
Tel . . . . Fax . . . . Email . . . .
Ragione Sociale . . . .
Codice fiscale . . . . Partita IVA . . . .

DATA . . . . FIRMA . . . . ☐ Fattura

FE 330

ISTRUZIONI PER L'ORDINE E SPESE DI SPEDIZIONE

Il modulo d'ordine dovrà essere compilato in tutte le sue parti ed inviato via posta o via fax ai recapiti indicati sul modulo d'ordine stesso. Gli ordini potranno essere fatti anche direttamente online dal sito www.elettroshop.com/ieshop. In questo caso non sarà necessario inviare il modulo d'ordine. Le spese di spedizione ammontano a euro 8,50 a cui vanno aggiunti euro 3,50 se si sceglie di pagare in contrassegno.

METODI DI PAGAMENTO

Si accettano pagamenti ino in contrassegno, carta di credit, bollettino postale o bonifico all'ordine. Per il contrassegno verrà applicata una spesa aggiuntiva di euro 3,50 per le spese di contrassegno. Forme diverse di pagamento devono essere previamente concordate.

Bonifico bancario

Appoggiato su Poste Italiane IBAN: IT 68 I 07601 01600 000070107552 intestato a Inware Edizioni srl.

Bollettino postale

Versamento sul conto corrente postale n. 70107552 intestato a Inware Edizioni srl. Inviare la ricevuta (o copia) del versamento.

Contrassegno

La merce verrà pagata direttamente al corriere alla consegna della merce. Il pagamento in contrassegno comporta l'addebito di euro 3,50 per spese di contrassegno.

Carta di Credito

Il pagamento con carta di credito può essere effettuato specificando i dati della carta via posta, email, web, fax o telefono. Per gli ordini effettuati sul sito il pagamento mediante carta di credito verrà effettuato attraverso una connessione sicura SSL per garantire la massima sicurezza.

TERMINI E CONDIZIONI

Tutti i prodotti sono coperti da garanzia. La garanzia copre tutti i difetti di fabbricazione entro un anno dal ricevimento della merce. Tutti i prodotti non funzionanti per uso improprio o incuria non saranno ritenuti in garanzia e saranno addebitati gli eventuali costi di riparazione. Tutti i prodotti verranno riparati e/o sostituiti direttamente dal produttore. Non sono coperti da garanzia i componenti elettronici (microprocessori, memorie, ecc.) La garanzia dei prodotti si intende F.co ns. sede, le eventuali spese di trasporto sono a carico del cliente salvo accordi diversi. Per dar corso alla riparazione/sostituzione in garanzia è necessario seguire l'apposita procedura di RMA.

PRIVACY

Ai sensi del Decr. Lgs. 196/2003 la informiamo che i dati trasmessi verranno impiegati coi principali scopi di indagini di mercato e nelle modalità previste dallo stesso, prevalentemente con mezzi informatici. Il conferimento, di norma facoltativo, è obbligatorio per permettere il rapporto commerciale. È in ogni caso fatto diritto dell'interessato esercitare i propri diritti, nei modi previsti dal "Titolo II art. 7" della legge sopra citata, scrivendo a Inware Edizioni srl, Via Giotto 7 – 20032 Cormano o tramite email all'indirizzo info@inwaredizioni.it

---

DIVENTARE MEMBRO INWARE EDIZIONI CONVIENE!

GENIUS MEMBERSHIP: • 11 nuovi numeri di Fare Elettronica sia in edizione cartacea sia in PDF (in coda all'eventuale abbonamento esistente) • 11 nuovi numeri della rivista digitale Firmware (in coda all'eventuale abbonamento esistente), tutto l'archivio delle riviste Firmware pubblicate mensilmente a partire da febbraio 2010 **+ GENIUS Bonus Pack**, ovvero: • accesso al CLUB della rivista Firmware con centinaia di articoli in PDF • accesso al CLUB della rivista FareElettronica (centinaia di articoli in PDF) • due ebook a scelta* (smartcard, CPLD, display, PIC, AmpOP, alimentatori, linguaggio C) • due raccolte delle annate di Fare Elettronica in PDF (dal 2003 al 2011) e Firmware (dal 2006 al 2011) a scelta*.

MAKER MEMBERSHIP: • 11 nuovi numeri di Fare Elettronica sia in edizione cartacea che in PDF (in coda all'eventuale abbonamento esistente) **+ MAKER Bonus Pack**, ovvero: • accesso al CLUB della rivista Fare Elettronica (centinaia di articoli in PDF) • un ebook a scelta* (smartcard, CPLD, display, PIC,AmpOP, alimentatori, linguaggio C) • una raccolta delle annate di Fare Elettronica in PDF a scelta* (dal 2003 al 2011).

SMART MEMBERSHIP: • 11 nuovi numeri della rivista digitale Firmware (in coda all'eventuale abbonamento esistente) • tutto l'archivio delle riviste Firmware pubblicate mensilmente a partire dal febbraio 2010 • 11 nuovi numeri della rivista digitale Firmware
(in coda all'eventuale abbonamento esistente) **+ SMART Bonus Pack**, ovvero: • accesso al CLUB della rivista Firmware con centinaia di articoli in PDF • un ebook a scelta* (smartcard, CPLD, display, PIC, AmpOP, alimentatori, linguaggio C) • una raccolta della annate di Firmware in PDF (dal 2006 al 2011) a scelta*

METODI DI PAGAMENTO

Bonifico bancario

Appoggiato su Poste Italiane IBAN: IT 68 I 07601 01600 000070107552 intestato a Inware Edizioni srl.

Bollettino postale

Versamento sul conto corrente postale n. 70107552 intestato a Inware Edizioni srl. Inviare la ricevuta (o copia) del versamento indicando nella causale: "Abbonamento Fare Elettronica"

Carta di Credito

Il pagamento con carta di credito può essere effettuato specificando i dati della carta via posta, email, web, fax o telefono. Per gli ordini effettuati sul sito il pagamento mediante carta di credito verrà effettuato attraverso una connessione sicura SSL per garantire la massima sicurezza.

Assegno bancario

È possibile spedire un assegno bancario insieme a questo coupon.

Paypal

Inviare il pagamento a paypal@invaredizioni.it.

Volete analizzate segnali misti a prezzi accessibili?
Niente compromessi sulla qualità o marchi sconosciuti?

Fidatevi di Teledyne LeCroy.

Oscilloscopi WaveAce 1000 & 2000

- Campionamento fino a 2 GS/s
- 32 misurazioni automatiche
- Memoria 1 Mpts/ch, 2 Mpts interallacciati
- Schermo a colori da 7" su tutti i modelli
- Connessioni USB for stampanti, chiavette e controllo remoto da PC
- Interfaccia multilingue (ITA)
- Ampia memoria interna per forme d'onda e settaggi

Vi serve un oscilloscopio potente e carico di strumenti per il collaudo e l'analisi? La serie WaveAce 1000 & 2000 di Teledyne LeCroy ve lo mette a dispozione. E con l'analizzatore logico LogicStudio, tramite l'uso di un qualunque PC, avrai anche la flessibiltà dell'analisi per segnali misti. Questa combinazione perfettamente integrata vi mette a disposizione un oscilloscopio potente e versatile che non comprometterà il vostro budget.

RS partnumber	Teledyne LeCroy partnumber	RS Italy Web Price	RS partnumber	Teledyne LeCroy partnumber	RS Italy Web Price
758-7843	WaveAce 1001	Euro 715	758-7860	WaveAce 2014	Euro 1800
758-7857	WaveAce 1002	Euro 920	758-7879	WaveAce 2022	Euro 1770
758-7850	WaveAce 1012	Euro 1150	758-7872	WaveAce 2024	Euro 2040
758-7854	WaveAce 2002	Euro 1130	758-7876	WaveAce 2032	Euro 2735
758-7863	WaveAce 2004	Euro 1196	758-7885	WaveAce 2034	Euro 2860
758-7866	WaveAce 2012	Euro 1415	714-1363	LogicStudio 16	Euro 1150

CODICE MIP 2829390

www.rs-components.it